(2)

TECHNICAL REPORT BRL-TR-3390

BRL

TECHNIQUES FOR EVALUATING LINE OF BEARING (LOB) SYSTEMS

ANDREW A. THOMPSON III
GARY L. DURFEE

SEPTEMBER 1992

U.S. ARMY LABORATORY COMMAND

BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

NOTICES

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1992	Final, August 1991–August 1992

4. TITLE AND SUBTITLE
Techniques for Evaluating Line of Bearing (LOB) Systems

5. FUNDING NUMBERS
PR: 1L162618AH80

6. AUTHOR(S)
Andrew A. Thompson III and Gary L. Durfee

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
BRL-TR-3390

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report consolidates some of the work done on the problem of locating objects or targets from line of bearing information. Line of bearing (LOB) or angle of arrival (AoA) information is determined by a location and a direction to a target. It is not possible to infer the location of a target from LOB information collected from one site. The simplest case is to infer the location of an object in a plane from two separate measurements. Errors in the measurements result in target location errors. First, through a discussion of errors, the critical features and sensitivities of LOB systems are treated. Following this, a technique for combining more than one estimate of a target location is discussed. Finally, a system that estimates the trajectory of a target from LOB estimates is examined. Both simulations and closed-form models are used to study LOB system performance.

14. SUBJECT TERMS	15. NUMBER OF PAGES
target detection, trajectory estimation, interferometers, line of bearing or angle of arrival	110
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

INTENTIONALLY LEFT BLANK.

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The authors would like to thank Charles Alexander, National Security Agency, Fort Mead, MD, for allowing us to use his memorandum "Error Elipses in DOA Measurements" as Appendix A. Also, we wish to thank Thomas Harkins, Richard McGee, Jerry Thomas, U.S. Army Ballistic Research Laboratory (BRL), and Charles Alexander, for reviewing this report and providing constructive comments.

INTENTIONALLY LEFT BLANK.

1. INTRODUCTION

This report consolidates some of the work done on the problem of locating objects or targets from line of bearing information. Line of bearing (LOB) or angle of arrival (AoA) information is determined by a location and a direction to a target. It is not possible to infer the location of a target from LOB information collected from one site. The simplest case is to infer the location of an object in a plane from two separate measurements. Errors in the measurements result in target location errors. First, through a discussion of errors, the critical features and sensitivities of LOB systems are treated. Following this, a technique for combining more than one estimate of a target location is discussed. Finally, a system that estimates the trajectory of a target from LOB estimates is examined. Both simulations and closed-form models are used to study LOB system performance.

The accuracy of the target location depends on the errors associated with LOB information and the special relationship between the target and the sensors. In discussing LOB systems, the baseline is the line segment connecting the two sensors, and the range is the distance from the center of the baseline to the target. LOB system errors include both sensor location errors and angle of arrival errors. AoA errors can be broken down into those associated with the sensor precision and accuracy, and those associated with the propagation of energy to the sensor. A standard measure of performance for these systems is a mathematical description of the LOB system's pattern of target location errors called the covariance. The sensitivity of the covariance of the target location with respect to system parameters will show the critical features of the system. Two external problems associated with these systems are cochannel interference and multipath. Cochannel interference occurs when another signal blends at the sensor with the signal of interest; this usually results in the reported angle of arrival not pointing at the source of either signal. Hutton (1984) discusses cochannel interference for interferometers. When the signal travels to the sensor there are usually several paths it can travel, these paths combine at the sensor creating the multipath error (Wilson 1968; Wallace 1979; Stratton, Wallace, and Bauerle 1991). Hutton and Alexander (1984) discusses the sources of error for amplitude comparison and interferometer LOB sensors. Skolnik (1980) discusses error models associated with specific radar signal processing methods. Other sources of error can be vibration and thermal distortions of the platform, the sensor housing, or the sensor elements. Navigational errors also influence performance; thus, the yaw, pitch, roll, and location errors associated with

the platform need to be specified. The evaluator of an LOB system must be aware of errors caused by external factors during intended operation as well as internal system errors.

2. BACKGROUND

There is a long history of interest in LOB problems stemming from the utilization of LOB information in surveying. Included among the investigators of these problems are Gauss and Laplace. Detailed investigations of LOB problems were undertaken during World War II motivated by military requirements for intelligence (INTEL). Since the war, new perspectives on LOB problems have appeared in the open literature. Both Koopman (1980) and Daniels (1951) give accounts of their WW II perspectives on the problem. Commenting on Daniels (1951), Professor E. H. Thompson says, "As a surveyor, I was extremely interested in Dr. Daniels paper. The subject illustrates how two groups of people working on the same lines may hardly have any contact with each other at all. Surveyors have been working on this problem for years, and it is very interesting to find the statistician taking up the surveyors problem with what is evidently a great deal of success." Wegner (1971) discusses this problem with a "least squares" perspective emphasized. Bjerhammar (1973) uses many LOB surveying problems as examples in his textbook. Alexander's work on this problem is included in the Geolocation Error Model; Bunts (1982) describes how to use this model. Grubbs (1964) gives a good description of measures of performance worthy of consideration for LOB systems; some of these are radial error, extreme spread, and circular error probable (CEP). The error distribution used to describe LOB location errors is the bivariate normal distribution. The error contours associated with a given probability are elliptical with the major axis in the direction of the range to the target. This paper will focus on errors in range as these are the dominant error source for most configurations. The models discussed find the major and minor axis for each system and can be converted to other measures of performance. The references herein give several perspectives for this problem. There are undoubtably other treatments worthy of mention, for instance Wiley (1985).

3. THEORY

In two-dimensional space, two spacially separate, nonparallel direction measurements can be combined into a target location. Finding the common point of two lines is not a problem. Describing the covariance ellipse associated with the location is sometimes troublesome. A

memorandum on this written by Charles Alexander is included as Appendix A. The reader who wishes to understand the mathematics associated with LOB uncertainties should read Dr. Alexander's explanation. Koopman uses a slightly different argument to reach the same conclusion. The program, LOBCA (based on Appendix A), is included as Appendix B. Using this program, the investigator can estimate the system accuracy for different geometries and LOB errors; thus, it is useful for parametric analysis. A method for finding the covariance associated with a set of observations follows.

The errors associated with a LOB measurement of a target are orthogonal to the line connecting the sensor to the target. For small angles, $\sin(\theta) \approx \theta$ when θ is in radians; thus, at a specified range, the magnitude of error at the target location is approximately equal to the angular measurement error in radians multiplied by the range to the target. The covariance of the location estimate can be found by associating each measurement with a variance and a direction in the space of reference variables (XY plane). The variance associated with each measurement is $R\theta$ where R is the distance to the target and θ is the angular error in radians. The diagonal matrix formed by the variances of the observations will be denoted by Σ. The direction associated with each measurement is orthogonal to the ray connecting the sensor to the target. The matrix formed by concatenating the normalized directions associated with each measurement will be denoted as X. Then the location error of a point by an LOB system will be $(X'\Sigma^{-1}X)^{-1}$ where the prime denotes transpose and the -1 denotes the inverse operator.

Each measurement imposes a restriction on the probable location of the target. It is the goal of the measurement process then to form a set of restrictions that constrain the target location in all possible directions. Directions that are not constrained will have large errors associated with them. From this, one can visualize that LOB measurements from the same general direction to the target leave the range direction unconstrained, and thus errors in range will be large. In many situations, small changes in the observations result in large changes in the location estimate, mathematically this instability is reflected by the constraints being represented by an ill-conditioned matrix, and this is directly attributable to the situational geometry. One technique to determine the stability of a matrix is to observe the magnitude of the determinant of the matrix; if it is small, then the problem is ill-conditioned.

For the evaluation of a specific system, both propagation errors and processing errors need to be considered. By relating system specific parameters like thermal noise or degrees off-boresight to a aggregate LOB error, the evaluator can quantify various features of the equipment performance in terms of specific missions. One very useful characterization of a received signal is its signal-to-noise ratio. As the signal passes through the antenna, to the receiver, and then to the processor, this ratio changes. The final LOB error is a function of this ratio.

4. SIMULATIONS

Simulations can also be used to evaluate system performance. It is always a good practice to develop simulations so that a section of code reflects actual system parameters. If this onto relationship between the system and code is preserved throughout development, then it is relatively easy to update the simulation to reflect changes or alterations of the existing concept. For a LOB system, most simulations include the sensor location errors and LOB measurement errors. These errors directly affect the perceived target location. By repeating the simulation for different errors, a set of perceived target locations is created. Measures of system performance are based on the covariance structure of the perceived target location. There are several general types of LOB simulations; these are discussed in the remainder of this section. The basic approach involves the following steps:

1. Study the problem and identify sources of variation.

2. Find a relationship between the dependent and independent variables.

3. Attach probability distributions to the error sources and run the desired number of replications for each combination of variables.

4. Define desired statistics or measures of performance.

5. If desired, tests can be performed to map the response surface.

The simulations RANLOB and AZEL were both designed under the above guidelines. Both use angle measurements that include errors to estimate the location of targets on the ground. RANLOB represents two observers using line of bearing measurements to locate a target. For convenience, observers can be located on the Y-axis and the target can be located on the positive X-axis. Errors in self-location and angle measurement are considered. The AZEL model represents a single observer taking azimuth and elevation measurements to locate a target. For convenience, place the observer above the Y-axis and locate the target on the positive X-axis. As in the previous model, errors in self-location and angle measurement are transferred to the perceived target location. Both of these programs use the same subroutine to calculate the summary statistics. The details of these programs are straightforward and can be observed by examination of the code. (See Appendices C and D, respectively.)

In many cases, the feature of interest is the trajectory of the target. The type of model selected to represent a target depends on the expected target dynamics. In estimating the path of a projectile over the last portion of its flight, a straight-line, constant-velocity trajectory is usually assumed. Estimation techniques based on this assumption can be tested and compared to those that assume more complex paths. Through the use of a simulation, estimation techniques can be compared and evaluated for various noise processes. Appendix E contains an overview of LOB2D, a simulation designed to consider these factors.

A highly detailed model that can evaluate systems using LOB information is the Geolocation Error Model (GEM). This closed-form model includes cochannel interference and many receiver parameters. It is designed to evaluate the performance of airborne platforms locating ground based emitters. Spread spectrum techniques and high-frequency models have been developed for this model (Alexander 1991). In addition to LOB information, time difference of arrival and differential doppler information can be utilized. Alexander (1980, 1982) contains a discussion of some of the theory contained in GEM.

Law and Kelton (1982) provide a readable text explaining some of the most important ideas in simulation modeling. The rationale for varying the random number seed is provided. Their discussion of random number generators includes algorithms for simulating specific distributions. When these distributions are available it is possible to examine the sensitivity of the selected measures of performance to different error distributions. Appendix F includes code for some

random number generators. Their text also mentions some potential problems associated with the interpretation of simulation results.

5. GEOMETRY

The geometric relationship between the sensors and the target needs to be mentioned when discussing system performance. This can be addressed by an assessment of system performance at a set of locations or through a measure of the situational geometry. One measure of this is the range-to-baseline ratio. This descriptor is particularly useful, as it's meaning is clear and easy to visualize. A more precise measure of the physical relationship is the angle formed by the rays connecting the target to each sensor. This angle descriptor captures the off-boresight situation more accurately than the previous one. At longer ranges, these descriptors become equivalent as the tangent of the angle approaches the radial measure of the angle.

When the angle formed by the line segments connecting the sensors to the target is small, measurement errors can have a drastic effect on the estimate of the target location. Figure 1 shows the geometry for a simple, two-dimensional LOB system. In this figure, S1 and S2 represent two LOB sensors separated by two units. These sensors are viewing a target at a distance of four units giving a range-to-baseline ratio of 2:1. The true lines of bearing from S1 and S2 intersect at the "True Target Location." On both sides of the bearing lines from S1 and S2 are drawn rays (dotted lines) which bracket hypothetical angular excursions between ±3.0°. The area formed by the intersection of the four angular excursion rays from S1 and S2 provide an example of the extreme area in which the target might be observed (the shaded area in Figure 1). It should be noted that the range axis of the shaded area is significantly greater than the cross-range axis and that the range axis is greater than two units long. A target located at four units might appear to be anywhere from three to more than five units away. Additionally, if the ±3° angular excursions were actually one standard deviation (normal distribution assumed), then ~68% of the time the target would be observed within the shaded area in Figure 1 (~32% outside the shaded area).

Figure 2 shows the result of relocating the sensors from S1 to S3 and from S2 to S4, giving a sensor separation of one unit. This results in a range-to-baseline ratio of 4:1. In this figure, the "Target Location Area" from S1 and S2 (two-unit baseline separation) of Figure 1 is displayed

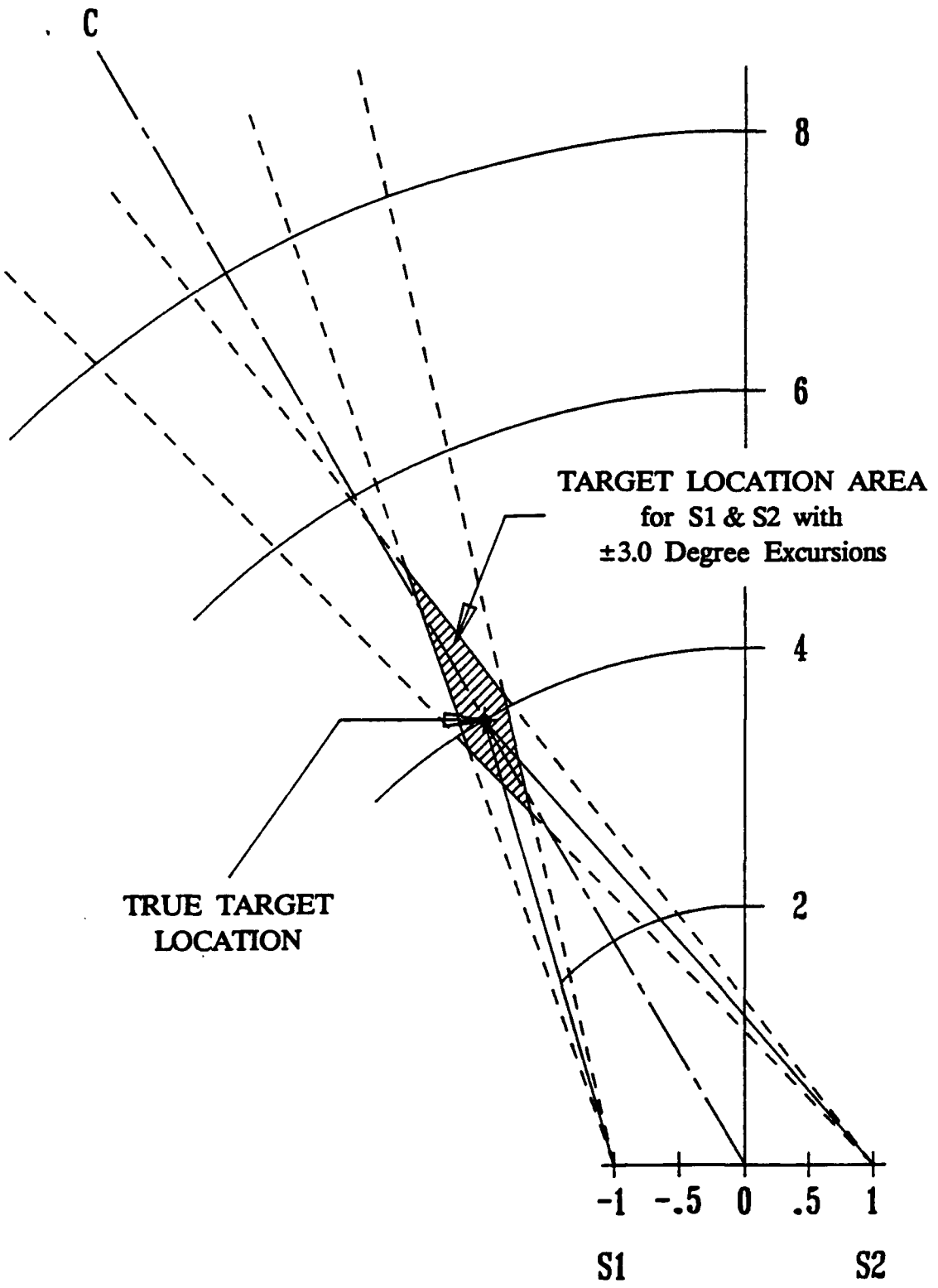

Figure 1. Two Dimensional LOB System Geometry with 2:1 Range-to-Baseline Ratio

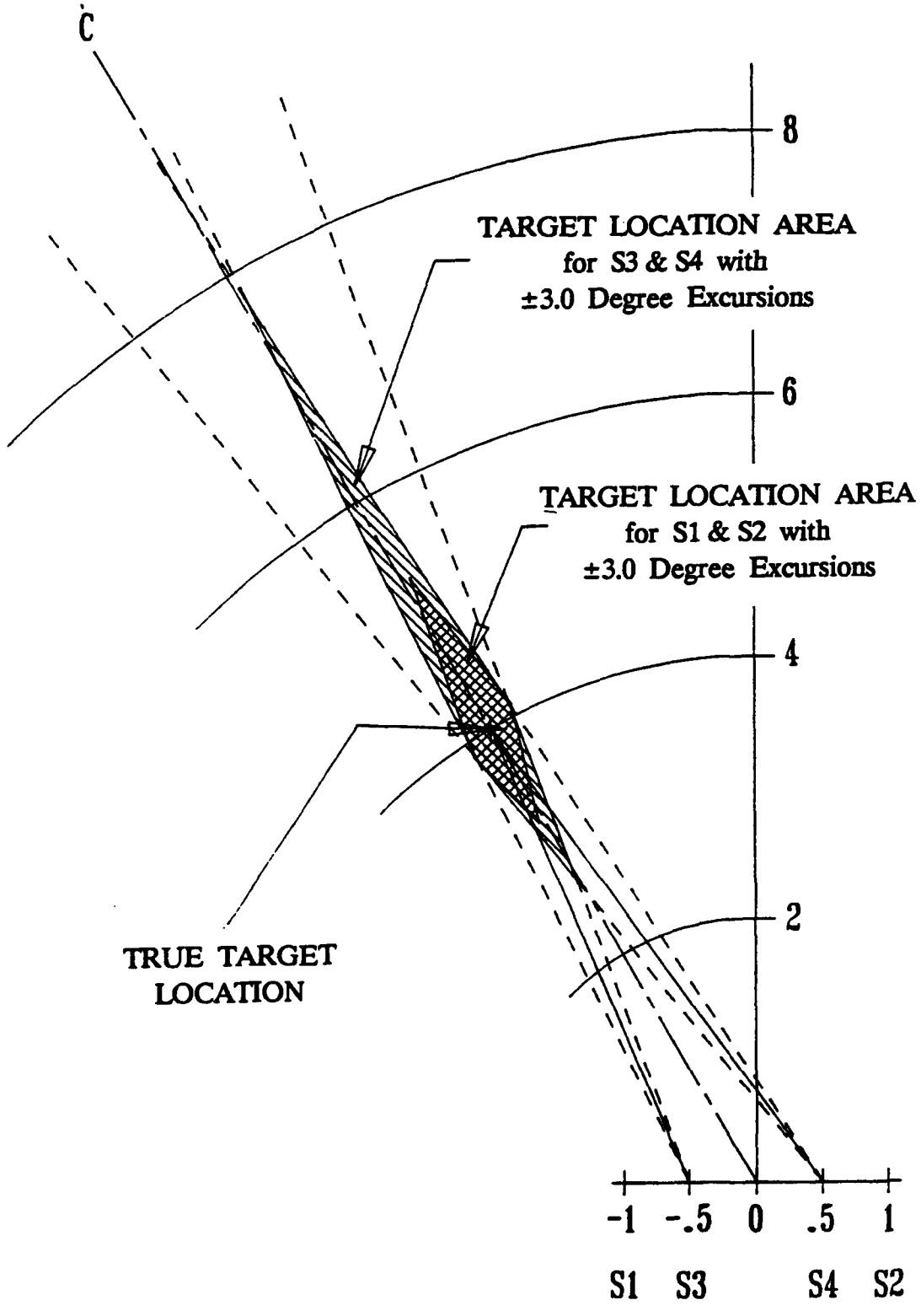

Figure 2. Two Dimensional LOB System Geometry with 4:1 Range-to-Baseline Ratio

as the darker cross-hatched area while the "Target Location Area" for S3 and S4 (one unit baseline separation) is displayed as the lighter cross-hatched area. It will be noted in Figure 2 that halving the sensor separation has very little effect on the cross-range axis. However, the reduced sensor separation has caused a large increase in the length of the range axis. In this case, the extreme spread of the range excursions is now more than 5 units long and the target, at a true range of 4, will have extreme range excursions of 2.5 and 7.7 units.

Further examples of the effects of LOB system geometry on range estimates for various angular excursions are plotted in the next four figures where graphs have been used rather than drawings so that larger range-to-baseline ratios can be presented. The effects of changing range-to-baseline ratio on extreme range excursions are plotted in Figures 3 and 4 for small angular excursions of 0.05°, 0.10°, and 0.50°. In these figures, negative extreme range excursion numbers indicate an observed target location nearer than the true location and positive extreme range excursion numbers indicate a observed target location further than the true location.

Figure 3 shows an on-boresight 0° attack azimuth. Here, for example, LOB angular excursions of ±0.5° could cause a target at a true range of 16 units to appear to be anywhere from 3.7 units closer to 6.1 units further than it's actual range. In general, as the range-to-baseline ratio increases or the angular excursions increase, the extreme range excursions about the true target location get larger.

Figure 4 shows an off-boresight 40° attack azimuth. Here, for the true target location of 16 units mentioned above, the target might appear anywhere from 4.3 units closer to 9.2 units further than it's actual range. Comparing these results with those from Figure 1, it will be noted that the off-boresight condition increases the extreme range excursions.

In some cases, the estimate of target location will be behind the baseline. This is seen by considering the two LOBs to a distant target and visualizing a rotation to the right of the right line (rotate right dashed line to right of S2 in Figure 1) and likewise, a leftward rotation of the left line (rotate left dashed line to left of S1 in Figure 1). Increasing the angular excursions (dashed lines in Figure 1), decreasing the baseline separation of the sensors, or increasing the range-to-target distance can create conditions in which the measured target location can appear to be behind the

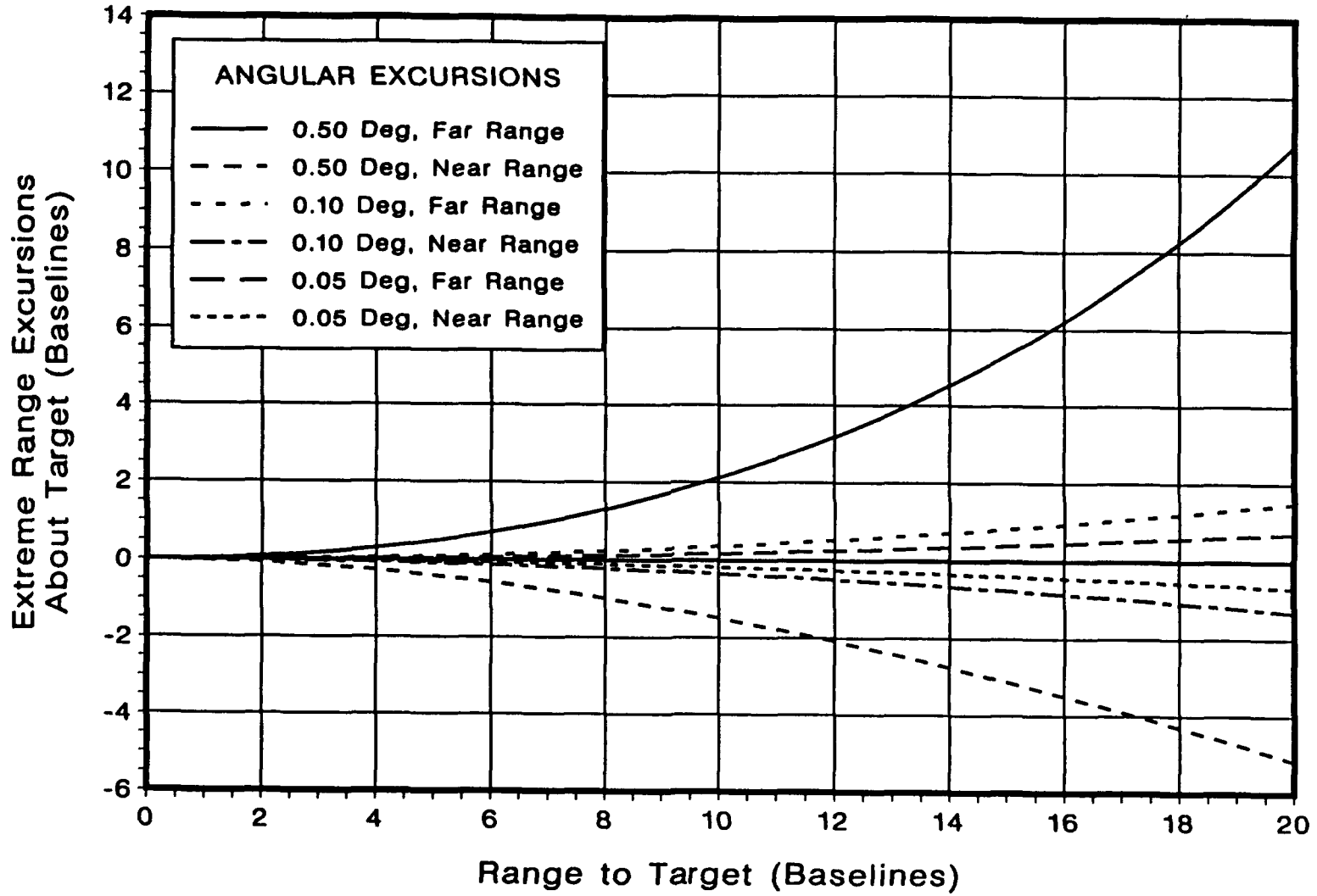

Figure 3. Geometric Range Excursions for a 0.05 - 0.50 Degree LOB System with a 1 Meter Baseline, Target On-Boresight

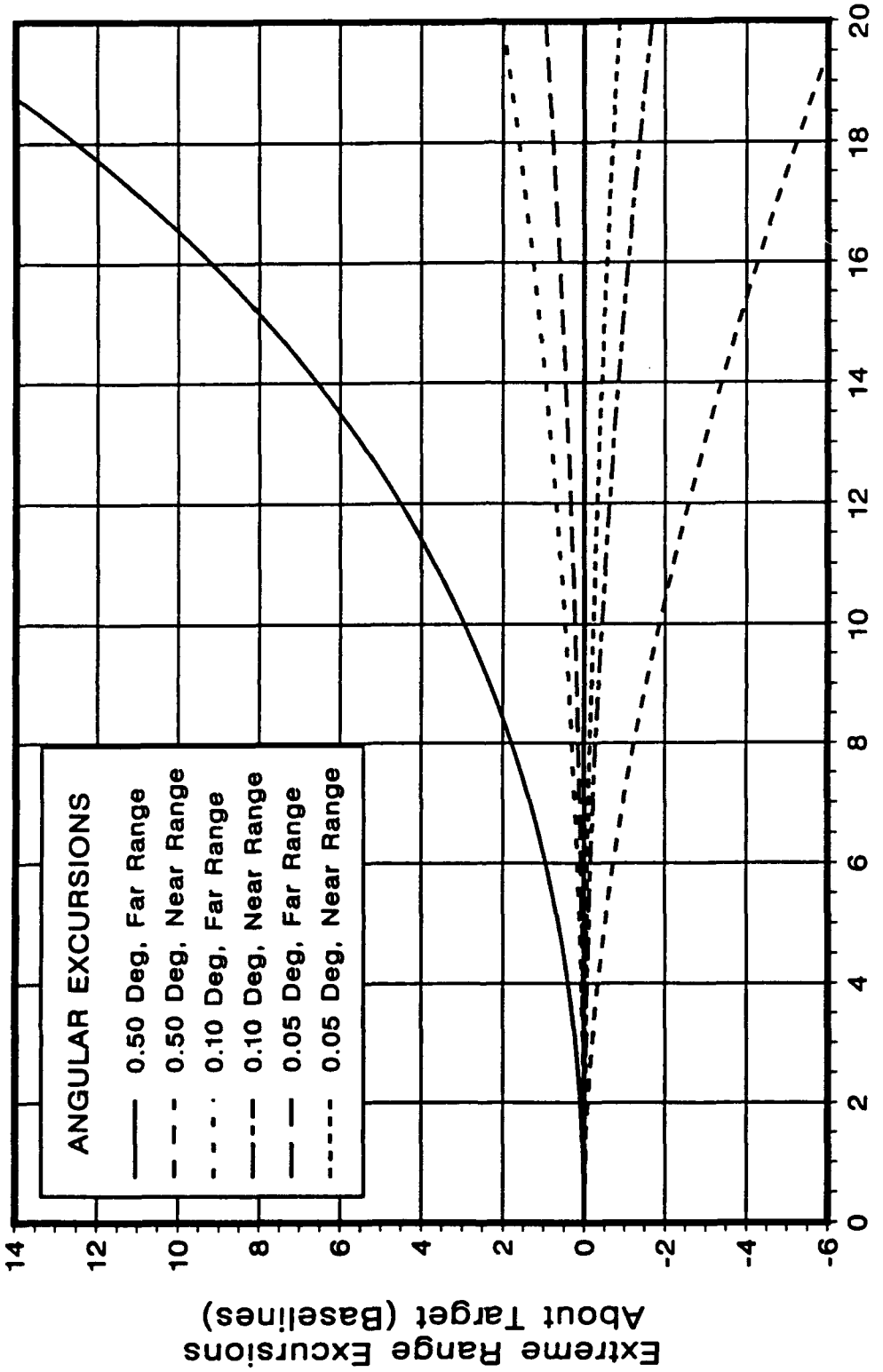

Figure 4. Geometric Range Excursions for a 0.05 - 0.50 Degree LOB System with a 1 Meter Baseline, Target 40 Degrees Off-Boresight

baseline. These effects can be seen in Figures 5 and 6 where the extreme range excursions are plotted as a function of range to target for angular excursions of 1°, 2°, and 3°.

In Figure 5, note that at a range of about 9.5 units for an angular excursion of 3° (14.5 units for angular excursion of 2°), the extreme range excursions go from a large positive value to a large negative value. This indicates that the two LOB sensor's bearing lines are crossing behind the baseline rather than in front of the baseline resulting in a target that appears to be behind the baseline.

Figure 6 is the plot for the 40° off-boresight attack azimuth. Here the target appears behind the baseline at about 7.4 and 10.9 units for angular excursions of 3° and 2°, respectively. Comparing Figure 6 with Figure 5 indicates that off-boresight attacks can cause the target to appear behind the baseline at shorter ranges.

One can see that small changes in the measurement errors can lead to implausible location estimates. Systems processing LOB information should be designed to ignore these cases. The simulation, LOB2D (Appendix E), contains a test for such points. When considering an LOB system for a mission, the possible geometries should be investigated to see if the mission is feasible.

6. ILLUSTRATIONS

This section discusses several types of analyses of LOB systems. The first two subsections are concerned with stationary targets, while the latter three are concerned with linear target motion.

6.1 Single Fix Analysis.

As an illustration of some of the above methods, consider the problem of evaluating the performance of a system in estimating the location of a target from two separate LOB measurements. A cut or fix refers to an estimate of the target's location from two LOB measurements. This section discusses the accuracy of a single fix.

As a first step, the program, LOBCA (Appendix B), was used to get a feel for the magnitude of the single fix errors for systems of various accuracies. Table 1 shows the single fix errors

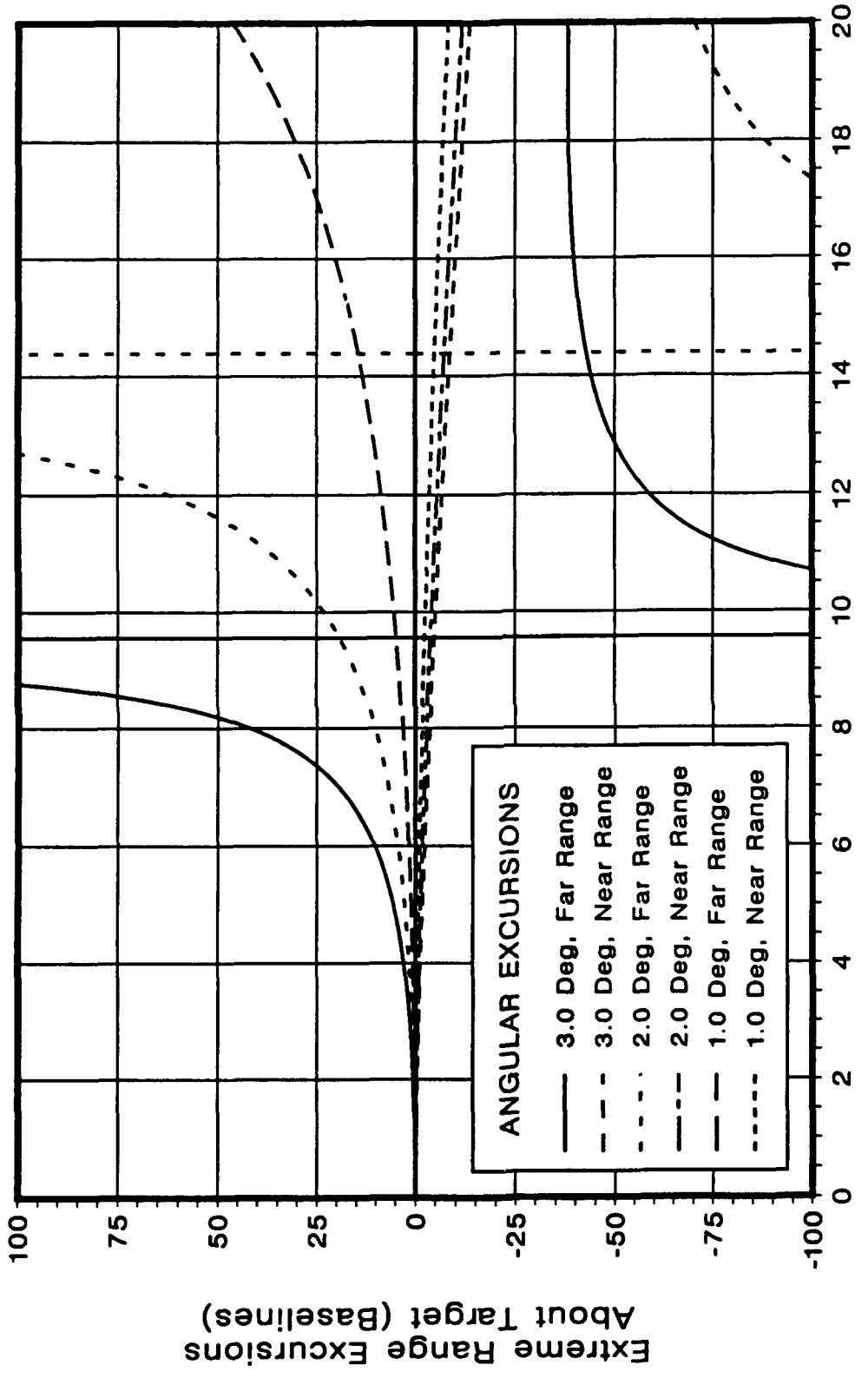

Figure 5. Geometric Range Excursions for a 1.0 - 3.0 Degree LOB System with a 1 Meter Baseline, Target On-Boresight

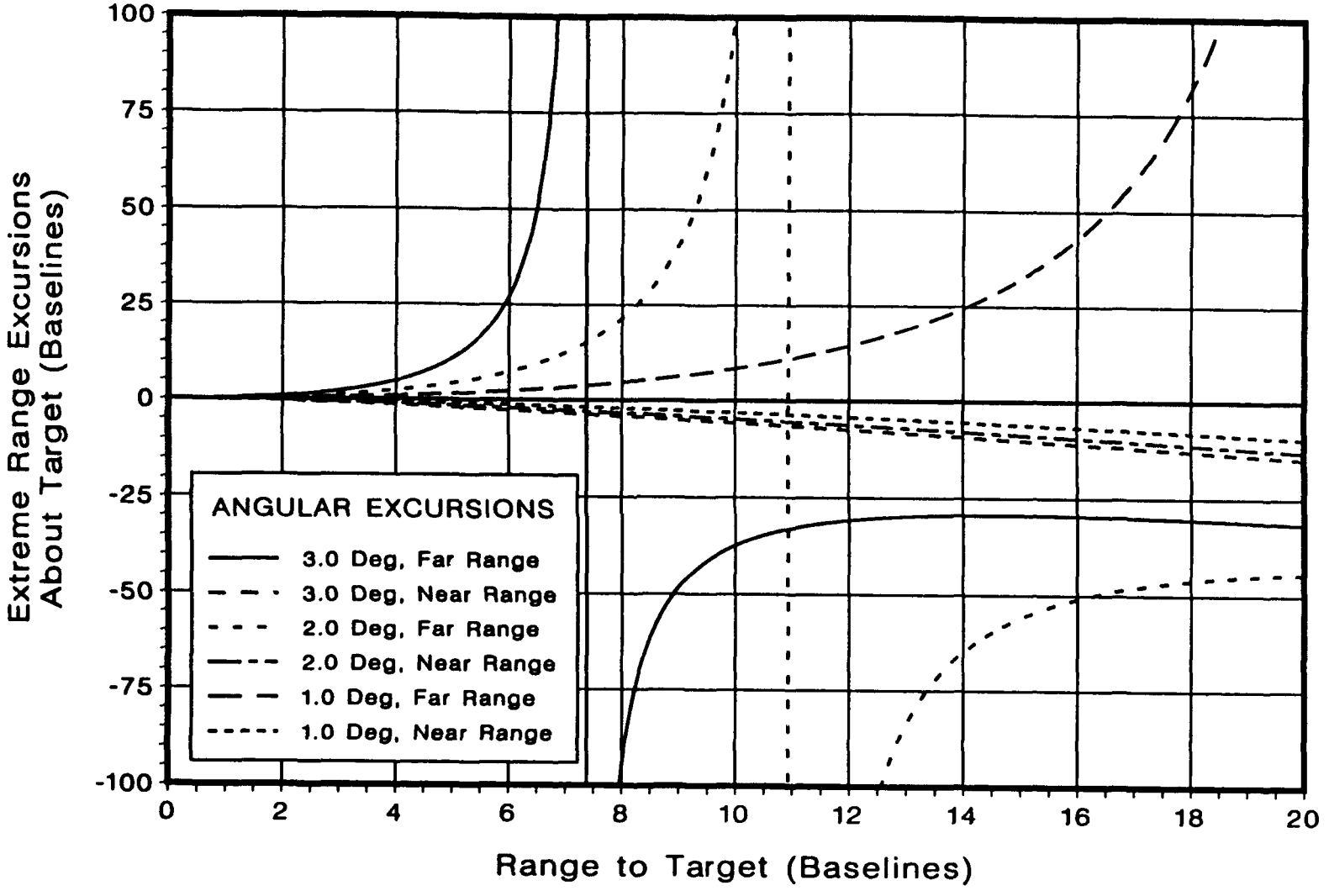

Figure 6. Geometric Range Excursions for a 1.0 - 3.0 Degree LOB System with a 1 Meter Baseline, Target 40 Degrees Off-Boresight

associated with the indicated conditions. The tabulated values are range standard deviations in baseline units. The boresight of a LOB system is the ray starting at the midpoint of the baseline and perpendicular to the baseline.

Table 1. Target on Boresight 1–3° Errors

Range	Angular Variations		
	1°	2°	3°
2 Baselines	.1138	.2098	.3147
3 Baselines	.2282	.4566	.6849
4 Baselines	.4011	.8022	1.2033
5 Baselines	.6232	1.2465	1.8697
6 Baselines	.8947	1.7895	2.6842
7 Baselines	1.2156	2.4312	3.6469

For Gaussian errors, 99% of the errors are within 2.81 standard deviation units of the mean, and 68% are within 1.0 standard deviation unit. Notice that for a 3° system with a target located seven baselines away, the estimated target position will have a standard deviation of half the range to the actual target position. (Table 1 is presented graphically in Figure 7.) When the target is off the system boresight, the performance is worse. The degradation can be thought of in two ways: first it can be attributed to a decrease in the angle formed by the sensors and the target; or it can thought of as due to the shortening of the baseline along the off-boresight ray. Table 2 shows results similar to Table 1 but for an off-boresight target. (Figure 8 displays this table as a graph.)

Table 2. Target 26.56° Off Boresight 1-3° Errors

Range	Angular Variations		
	1°	2°	3°
2 Baselines	.1154	.2308	.3462
3 Baselines	.2533	.5066	.7598
4 Baselines	.4464	.8928	1.3392
5 Baselines	.6948	1.3895	2.0843
6 Baselines	.9983	1.9966	2.9949
7 Baselines	1.3571	2.7141	4.0712

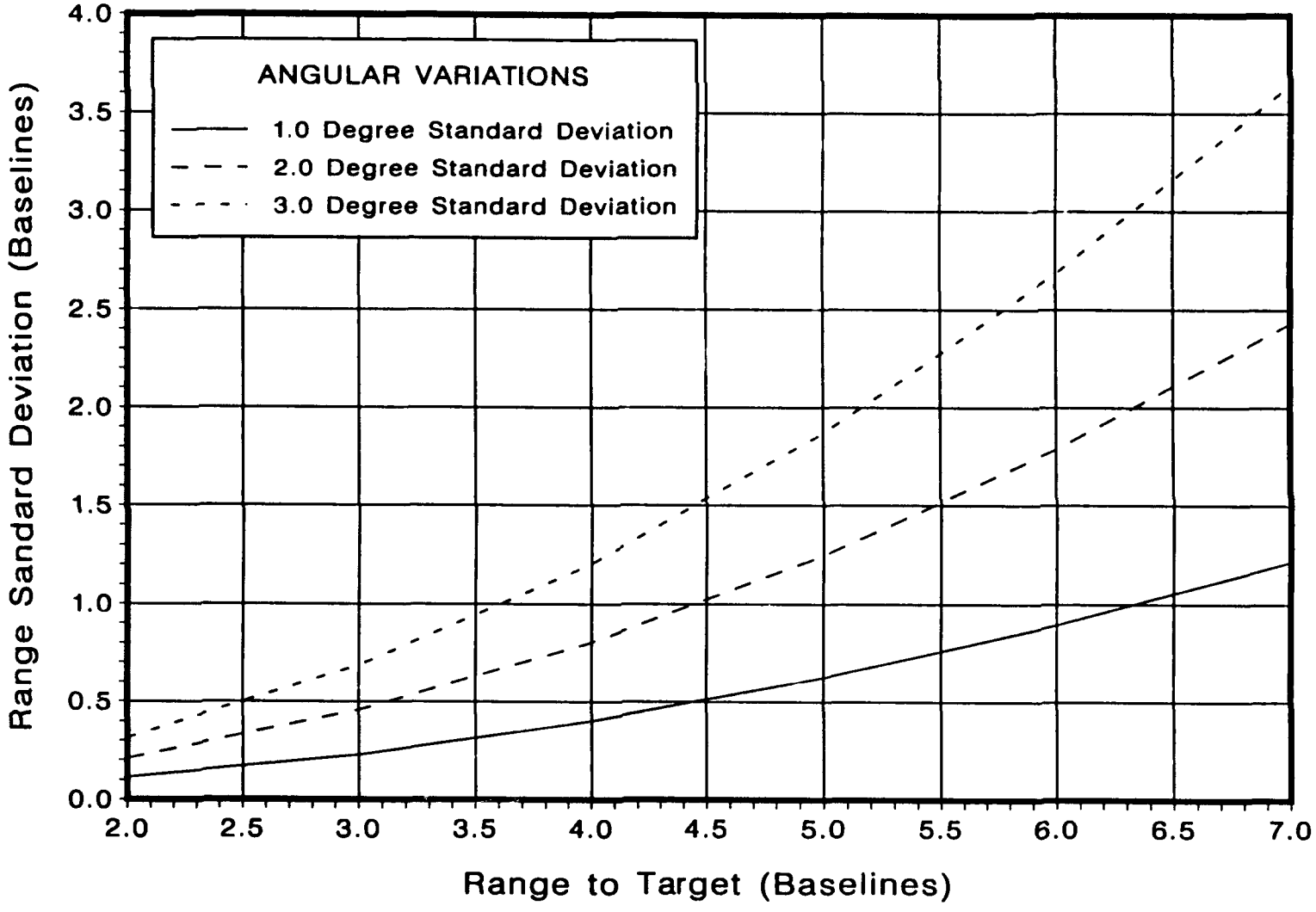

Figure 7. Standard Deviation of Range Error for a 1.0 - 3.0 Degree LOB System, Target On-Boresight

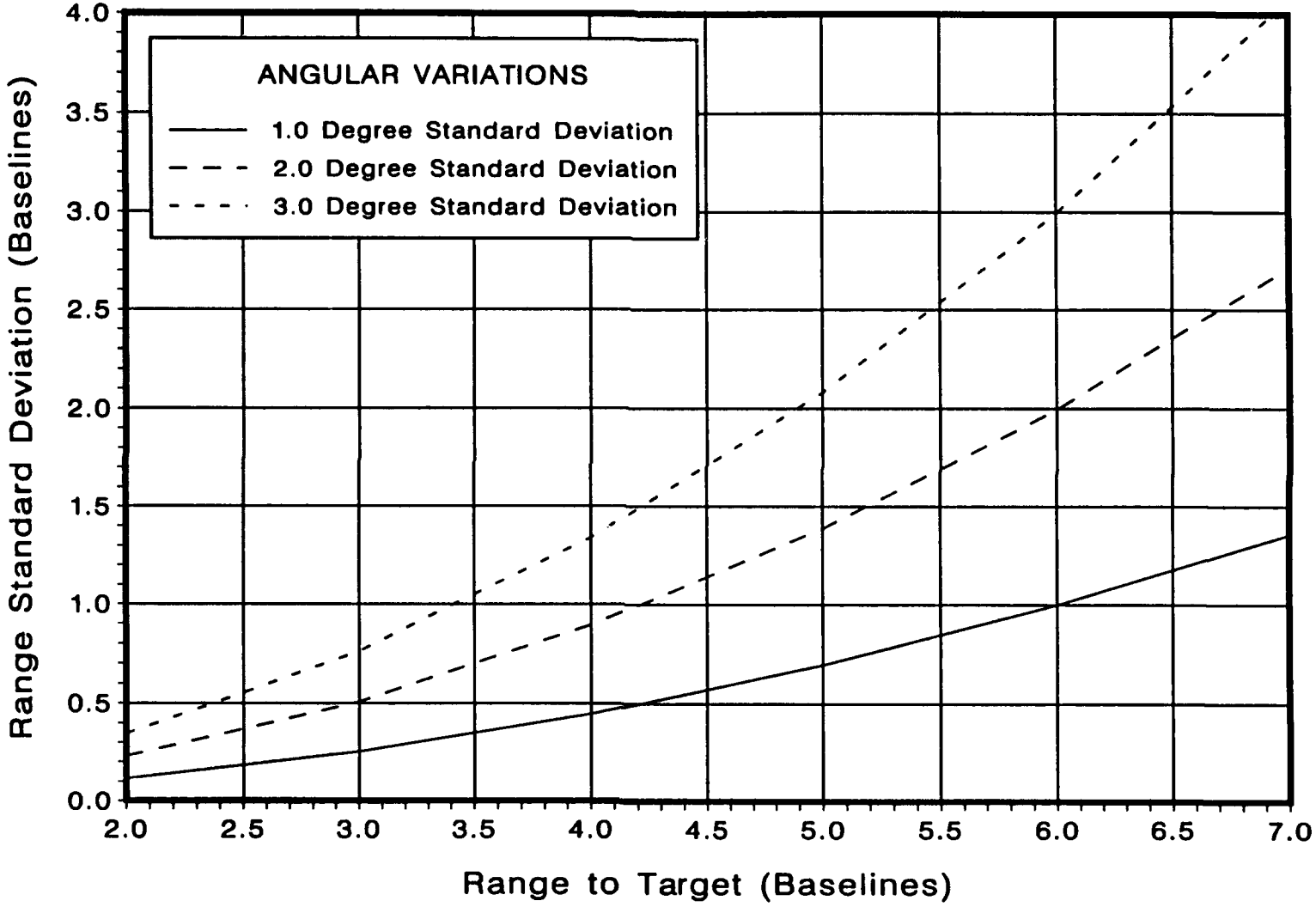

Figure 8. Standard Deviation of Range Error for a 1.0 - 3.0 Degree LOB System, Target 26.56 Degrees Off-Boresight

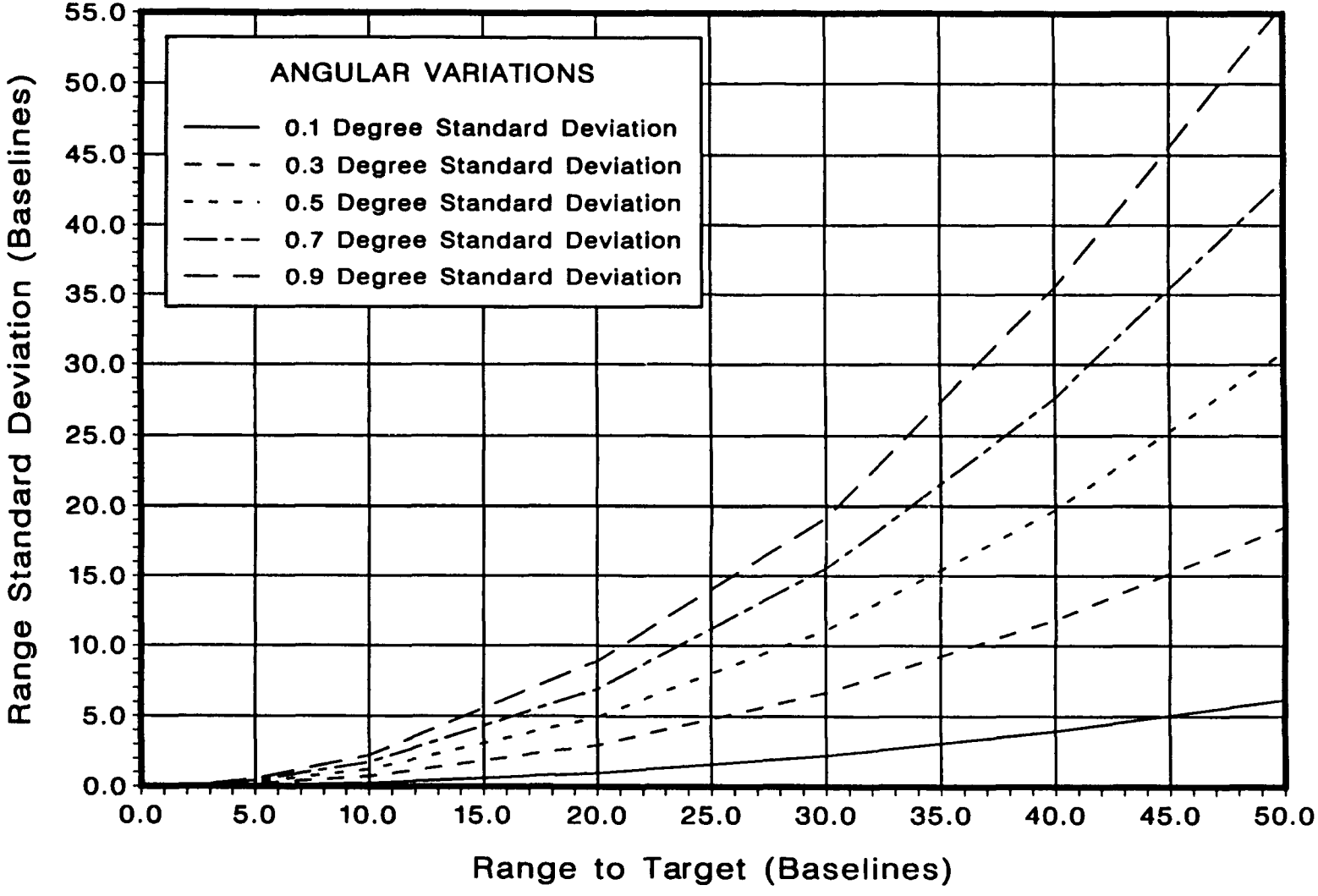

Figure 9. Standard Deviation of Range Error for a 0.1 - 0.9 Degree LOB System, Target On-Boresight

For a given range, the performance is always best along the system boresight. If the target is on the line that contains the sensors, then the range estimator will have infinite error. If θ is the angular measurement from the boresight to the target, then the error will be approximately proportional to product of the boresight error and $1/\cos\theta$.

Tables 1 and 2 include errors typically associated with electronic systems. Linear interpolation can be used to find values between the tabled values. As an example, consider a system with a 3-m baseline estimating the range to a target 10 m away. Since the range to the target is 3 1/3, the range error in baseline units for a 1° system is (1/3)(.4011-.2282) + .2282, or .2858. This is a range error of .8574 m. Table 3 and Figure 9 give the boresight performance of a system for small angular errors. Systems with errors in this range will typically operate in infrared or optical bands. The transit is an example of a device that can be used to make LOB measurements. Transits used for survey work have measurement errors in the range of 30 seconds or .00833°.

Table 3. Target on Boresight .1–.9° Errors

Range	Angular Variations				
	.1°	.3°	.5°	.7°	.9°
1 Baselines	.0031	.0093	.0156	.0216	.0278
3 Baselines	.0228	.0685	.1142	.1598	.2055
5 Baselines	.0623	.1820	.3116	.4362	.5609
10 Baselines	.2474	.7423	1.232	1.732	2.226
20 Baselines	.9879	2.963	4.939	6.915	8.891
30 Baselines	2.222	6.666	11.11	15.55	19.99
40 Baselines	3.950	11.85	19.75	27.64	35.54
50 Baselines	6.171	18.51	30.85	43.20	55.54

For a 0.9° system, estimating a target location 50 baseline units distant, the standard deviation of the range estimate will be greater than the range to the target. It may be possible to calibrate some equipment quickly by observing the point at which the standard deviation of the range estimate blows up. Table 4 contains some data for systems that are able to measure angles with less than 0.1° error. Figure 10 shows the exponential increase in the standard deviation as a function of range.

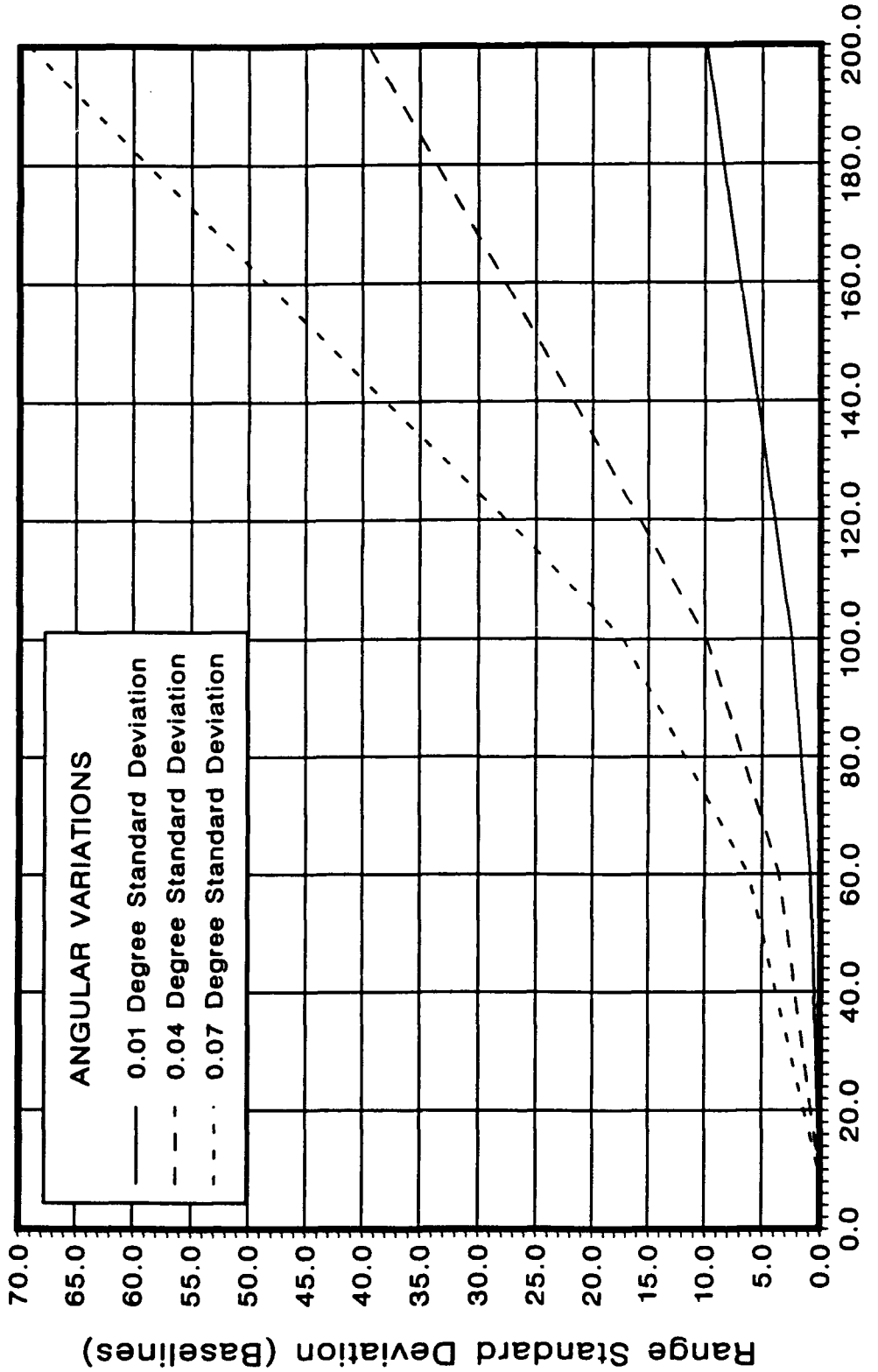

Figure 10. Standard Deviation of Range Error for a 0.01 - 0.07 Degree LOB System, Target On-Boresight

Table 4. Target on Boresight .01–.07° Errors

Range	Angular Variations		
	.01°	.04°	.07°
2 Baselines	.0011	.0042	.0073
10 Baselines	.0247	.0989	.1732
60 Baselines	.8886	3.554	6.220
100 Baselines	2.468	9.873	17.27
200 Baselines	9.873	39.49	69.11

As a general rule, these systems become useless as the angle between the sensors and the target approaches the measurement error. The task of the astronomer in mapping interstellar distances and plotting the location of galaxies is daunting. It is possible that some of their techniques may be useful to terrestrial LOB systems. It is possible to imagine a system in which various vehicles use their existing sighting systems to find LOB measurements to a distant object, then by two or more vehicles communicating these measurements, a positional fix of the target could be established and used for INTEL or targeting purposes. Typical fire control systems have accuracies of .006° and could locate distant objects accurately in most cases.

6.2 Multiple Cuts. In some situations, more than one system may be estimating the target location; additionally, there may be many location estimates from each system. This type of situation is discussed by Alexander (1980), Torrieri (1984), and Thompson (1991a). The procedure is to use recursive, weighted least-squares processing to combine the location estimates. (Appendix G includes one approach.) When processing more than one estimate, the possibility of correlated errors needs to be considered. Thompson (1992) discusses the problems correlated errors can cause when estimating the mean.

If the target is assumed to be moving, there are many reasonable combination schemes for the data. In some cases, it is reasonable to project the location estimates onto a parametric function that captures the target motion. The simplest case is if the target is known to be moving in a straight line. In this case, each value X or Y, can be modeled by a line using time as the independent variable. More complex trajectories require quadratic or higher order terms within the model. When the dynamics of the target need to be considered a state space filter is an appropriate estimator of the target trajectory.

6.3 Scenario. Consider the following scenario as a two-dimensional problem to investigate: There is an incoming projectile approaching a friendly target at a constant attack azimuth and a constant speed. The threat projectile can only be detected at close ranges, say 20 m or less. It is desired that a defeat mechanism intercept and destroy the incoming threat prior to impact, say at 10 m. The defense tracking system's job is to track the threat from the detection point (20 m), continually make predictions as to WHEN and WHERE the threat will cross the 10 m intercept line, and at some distance greater than the intercept distance (perhaps 11 m), make a final decision as to WHEN and WHERE intercept will occur.

For a 2-m sensor baseline separation, this scenario would result in a range-to-baseline ratio of 10:1 at the maximum detection range. This geometry would be difficult to represent in a drawing since a projectile at 20 m, with sensor angular excursions of only ±0.5°, would result in extreme range excursions of 18 and 23 m (from Figure 4) for an angle of attack of 40°. For angular excursions between ±2.0°, the projectile could appear to be from 16 to 118 m distant (from Figure 6). Such a drawing would be difficult to scale so that the areas of interest would be legible. Therefore, for the sake of illustration for the following discussion, Figures 11 and 12 are drawn for a range-to-baseline ratio of 2:1 for sensors with ±3.0° angular excursion.

A two-dimensional version of the LOB scenario is depicted geometrically in Figure 11 where S1 and S2 are the sensor locations, C-C is the projectile's trajectory, the True Projectile Location and Intercept Line are labeled, and the area in which the projectile might be located is shaded. The problem is then one of determining the location, direction of travel, and speed of the incoming projectile in order to predict it's crossing point (the WHERE) and time of arrival (the WHEN) at the intercept line. Two possible estimates of the projectile's location are shown as M1 and M2.

In order to simplify this problem to see the effects of errors in only range estimates, assume that the projectile's speed and angle-of-attack are exactly known. This reduces the problem to one of the defensive system's ability to estimate the projectile's range and cross-range location. In Figure 11, if the projectile's estimated location was at M1, then it's trajectory would cause it to cross the intercept line at a. Similarly, if the projectile's estimated location was at M2, then it would cross the intercept line at b. Since the true intercept should be at x, it can be seen that,

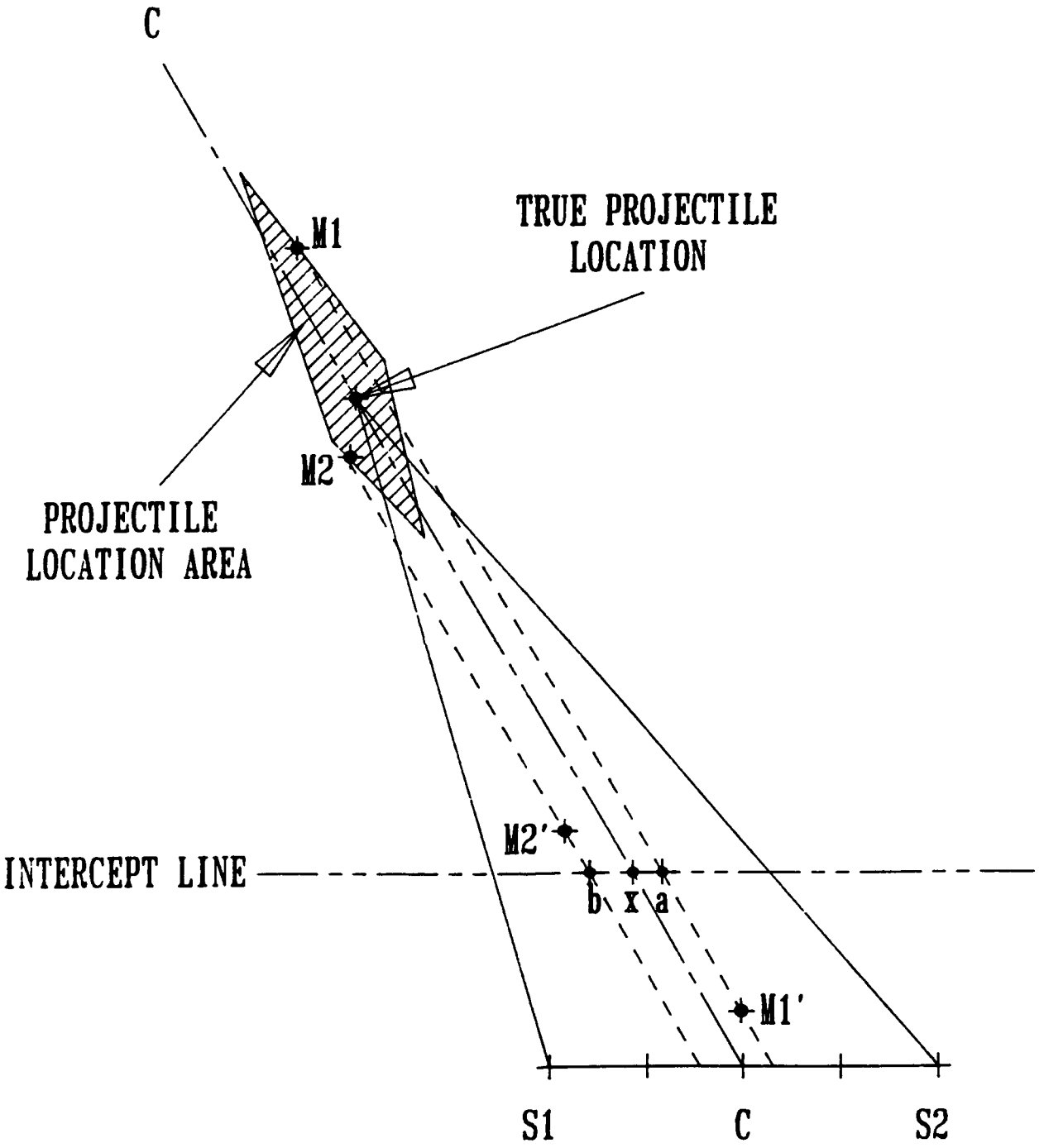

Figure 11. Two Dimensional Range Estimation Errors for LOB System with Known Projectile Speed and Angle-of-Attack

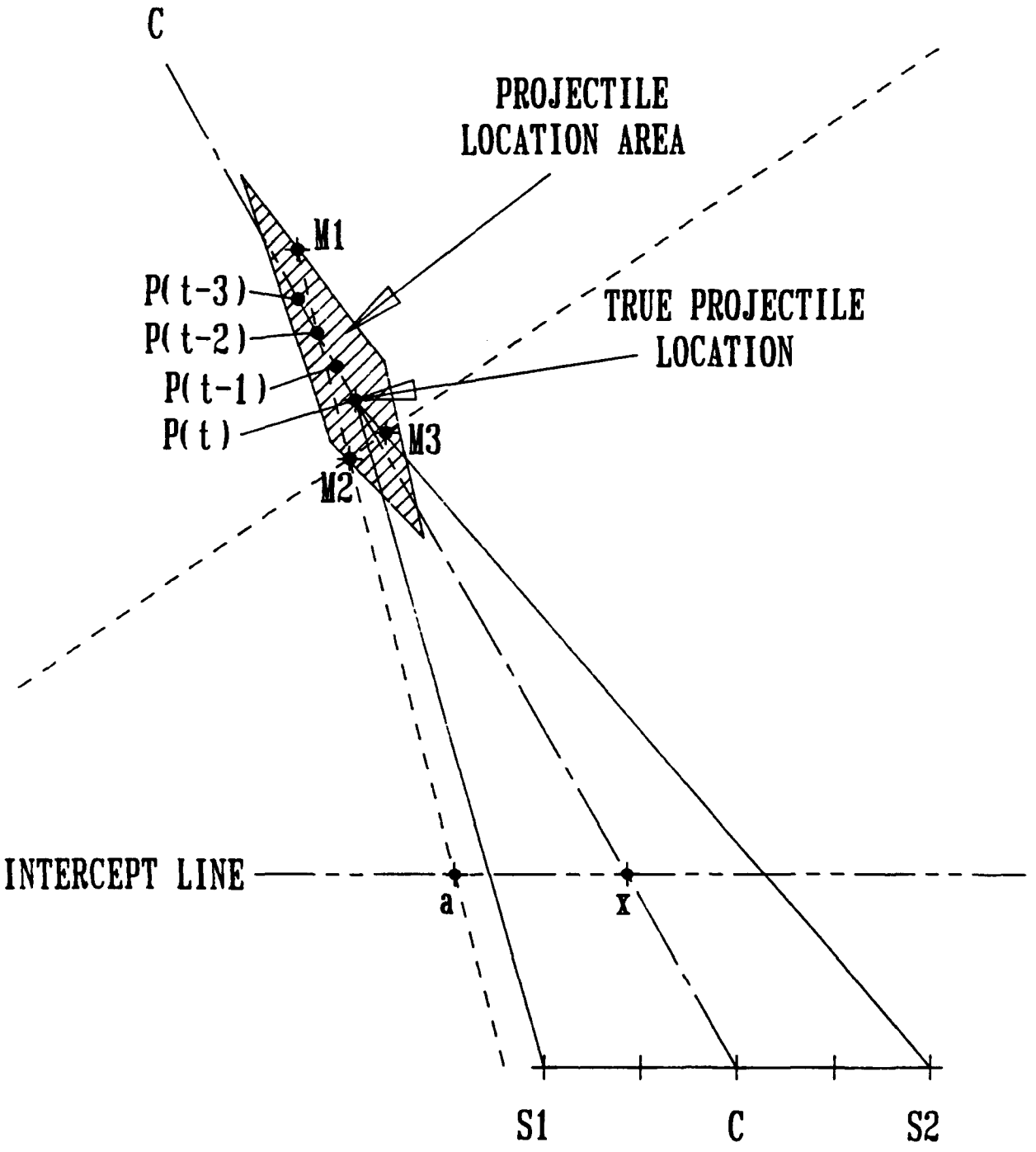

Figure 12. Range, Speed, and Angle-of-Attack Estimation Errors for LOB System

with the stated assumptions (known projectile speed and angle-of-attack), the error in WHERE the projectile will cross the intercept line is not large.

But WHEN will the projectile cross the intercept line? The much larger errors in estimating range will cause the true projectile location, apparently sighted at M1, to actually be at M1′ (well past the intercept line) when time-of-flight predictions indicate it should be at a on the intercept line. Similarly, if the estimate of the projectile location is at M2, then a time-of-flight prediction will put the projectile at b when it is actually at M2′ (short of the intercept line). Estimates of range that are longer than the true range result in the projectile's true location being closer than expected.

In this scenario, a defensive system that could exactly predict a projectile's speed and AoA would give a reasonable estimate of WHERE the projectile would cross the intercept line, but would have considerable trouble in predicting WHEN it would cross the intercept line.

Now consider a defensive system which is unable to exactly predict an incoming projectile's velocity and AoA but must determine these based on successive estimates of the projectile's location. Such a two-dimensional system is shown in Figure 12 where the True Projectile Location, the Projectile Location Area, and the Intercept Line are labeled. In addition, M1, M2, and M3 are possible estimates of the projectile's location at some time step and P(t), P(t-1), P(t-2), P(t-3) are the true locations of the projectile at the current time step, the previous time step, two time steps ago, and three time steps ago, respectively.

Consider that at the current time step, t, the projectile appears to be at M2 and at the previous time step, t-1, the projectile appeared to be at M1. Simply connecting M1 and M2 (dotted line) to determine the AoA would result in the projectile appearing to intersect the Intercept Line at a, to the left of the actual intercept which is at X. Also consider that if M2 was the current location estimate and M3 was the previous estimate, then the estimated projectile intercept would be much further to the left of the actual intercept X. Finally, consider that if M3 was the current location estimate and M2 was the previous estimate, then the projectile would appear to be moving away from the intercept line.

In order to reduce the possible large variances in AoA estimates and the possibility of a current location estimate at a range greater than the previous estimate, it would be desirable to sample less frequently so that subsequent samples do not fall within the Projectile Location Area of the previous sample. This is, however, statistically not practical. The geometry in Figure 12 is for a range-to-baseline ratio of 2:1. As was seen in Figure 2, increasing this ratio greatly increases the variations in range estimates (the length of the "Projectile Location Area" increases). Therefore, if the sampling rate is decreased in order to reduce or eliminate overlapping Projectile Location Areas, the result may be that there are very few samples from which to make a prediction.

Another problem with the scenario depicted in Figure 12 is the difficulty in predicting the projectile's speed from successive location estimates. Again, if M2 is the current location estimate at time t and M1 is the previous estimate at time t-1, and the sampling time step is known, then the projectile's speed could be calculated based as the distance between M1 and M2 divided by the time step. However, as can be seen from Figure 12, the distance from M1 to M2 (estimated) is much greater than the true distance traveled, P(t-1) to P(t). This will cause large errors in estimating projectile speed which will result in large errors in determining WHEN the projectile will cross the intercept line. Estimates of projectile speed which are too high will result in an estimated projectile arrival at the intercept line that is well ahead of the actual arrival.

In order to produce an effective system for determining both the WHEN and WHERE a threat projectile will cross an intercept line, consideration must be given to the choice of methods selected for making the location estimates and the way in which individual observations are combined to produce the desired predictions. A connect-the-dots method may result in large prediction errors. A least-squares recursive estimation method can be used to combine target location observations in an attempt to reduce trajectory estimation errors. Least-squares estimators are optimal for identically distributed, independent errors. (Some least-squares estimation techniques are discussed in Appendix G.)

6.4 A Simulation Study of the Scenario. Consider again the scenario stated in the first paragraph of Section 6.3. There is an incoming projectile approaching a friendly target at a constant attack azimuth and a constant speed. The threat projectile can only be detected at close ranges. The problem is to track the threat from the detection point, continually make predictions

as to WHEN and WHERE the threat will cross a predetermined intercept line, and at some distance prior to the projectile reaching the intercept line (to allow for the function of an appropriate intercept mechanism), make a final decision as to WHEN and WHERE intercept will occur.

In order to parametrically evaluate such a scenario, the two-dimensional Monte Carlo type simulation model, LOB2D, was developed to provide estimates of WHEN and WHERE a projectile will cross a predetermined intercept line as a function of various input parameters. These parameters specify important features of the projectile (speed, attack azimuth, detection range, aim point), the sensor (baseline separation, one standard deviation angular variations), the geometry (intercept line location, decision line location), and the processor (sampling rate).

In addition, LOB2D also allows for setting various cutoff parameters so that if an estimate of the projectile's location is either too close (i.e., behind the sensor baseline), too far (beyond possible detection range), results in a predicted impact location too far to the left or right of the baseline to be a threat, or produces a projectile speed estimate which is unreasonably high, then the estimate is discarded. In LOB2D, angular errors are simulated by independent random draws from a normal distribution with a specified standard deviation. Angular variations are about the true bearing line from the sensor to the projectile. Acceptable location estimates are combined using a recursive least-squares estimation technique (presented in equations 4, 5, and 6 of Appendix G) which estimates the projectile's start location, current location, speed, and angle-of-attack. The least-squares routine uses equal weights for all locations that are not discarded. (A summary discussion of the LOB2D simulation model, including sample inputs and outputs, can be found in Appendix E.)

Although LOB2D provides information on both the along-intercept-line position mean (bias) and standard deviation (the WHERE) and the range-to-intercept line position mean (bias) and standard deviation (the WHEN), only range-to-intercept line sample results will be presented in this report. Figures 13 through 16 provide LOB2D sample results for a projectile approaching at a constant speed of 300 m/s with detection at a range of 20 m. The intercept line is located at 10 m, the decision line is located at 11 m, and there are two sensors separated by a baseline distance of 2 m. The sensor angular variations are based on random draws from a normal distribution with one standard deviation values ranging from 0° to 3° in 0.02° steps.

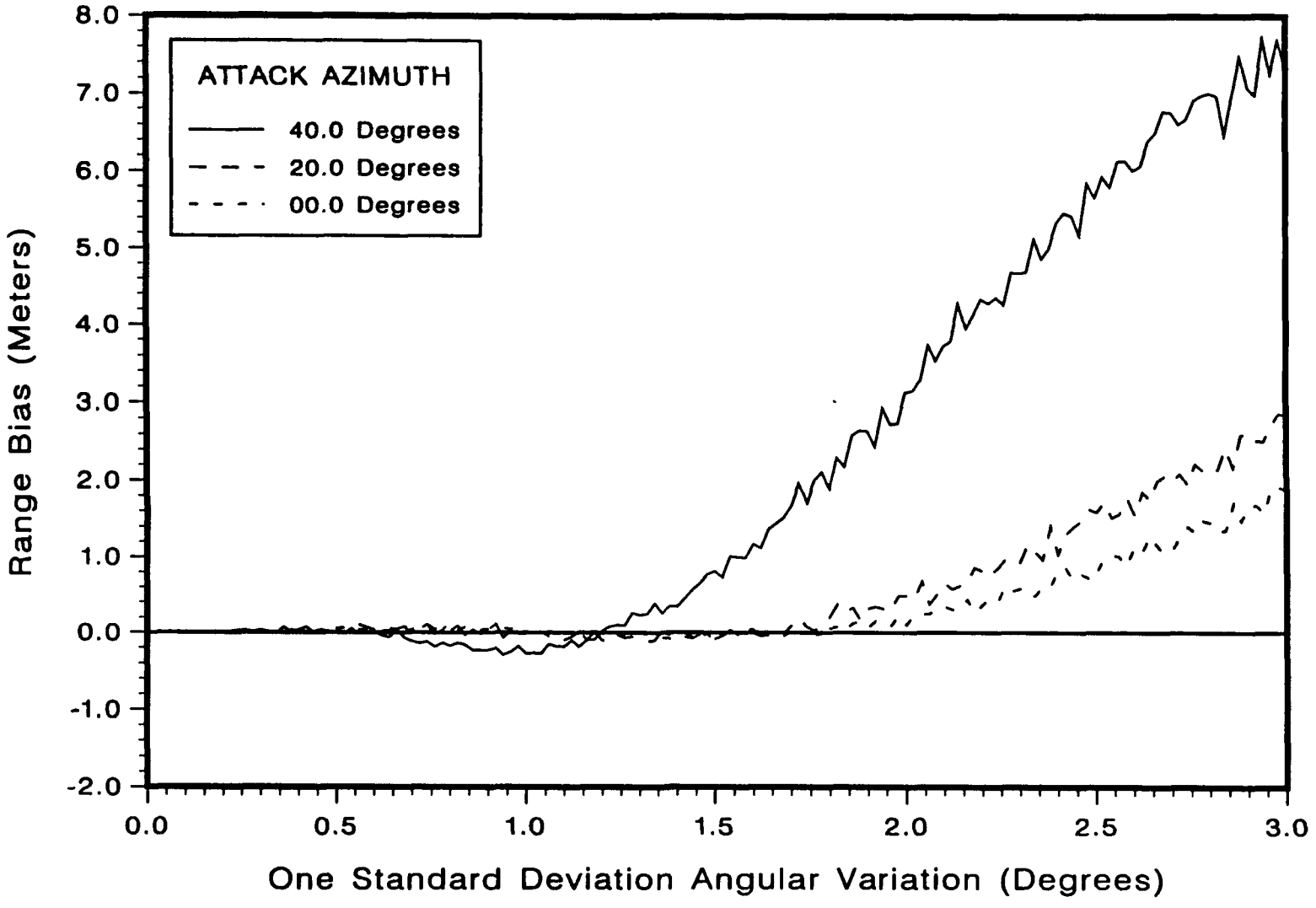

Figure 13. Range Bias for LOB Tracker Simulation with Angular Variations from 0.0 to 3.0 Degrees

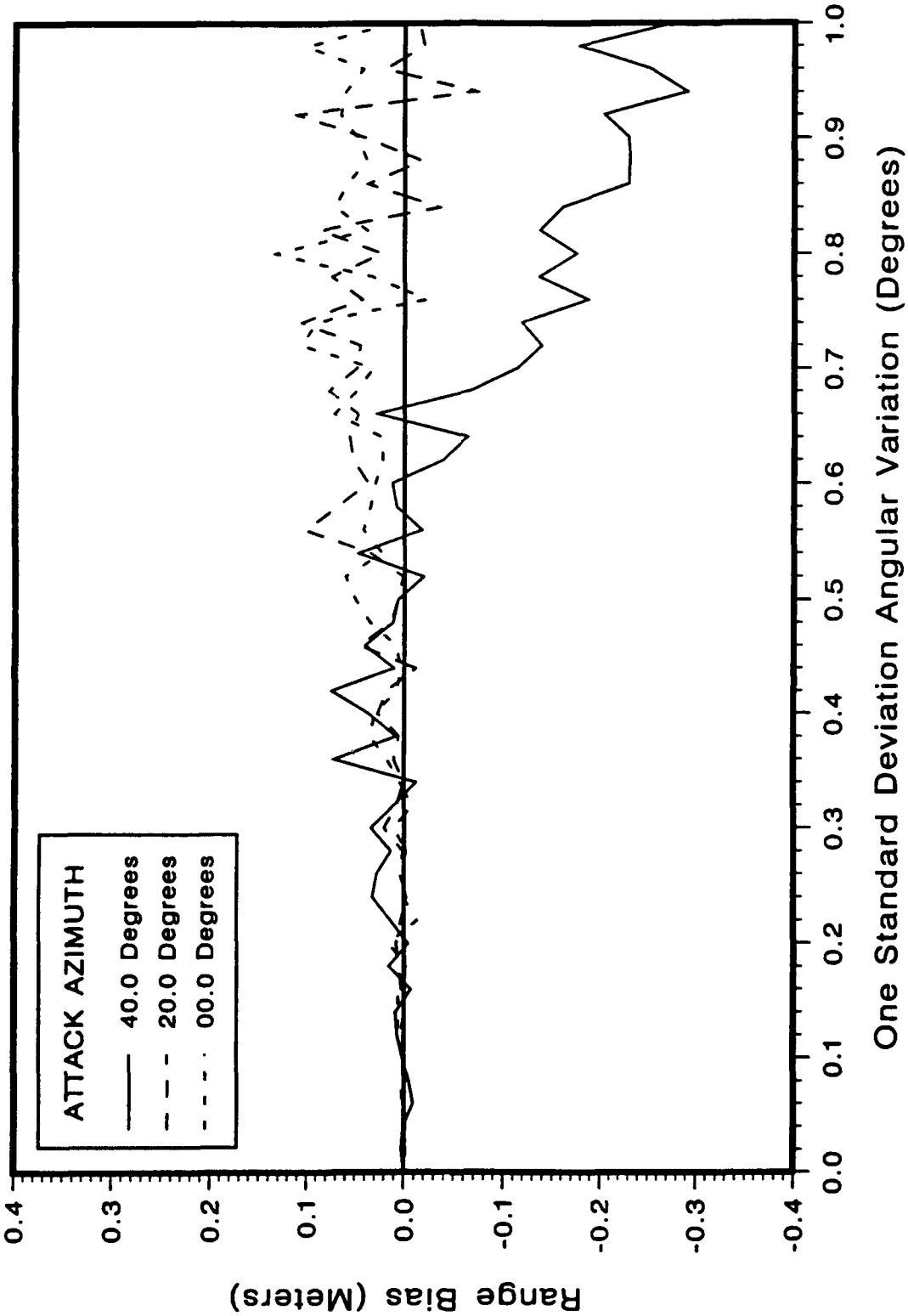

Figure 14. Range Bias for LOB Tracker Simulation with Angular Variations from 0.0 to 1.0 Degrees

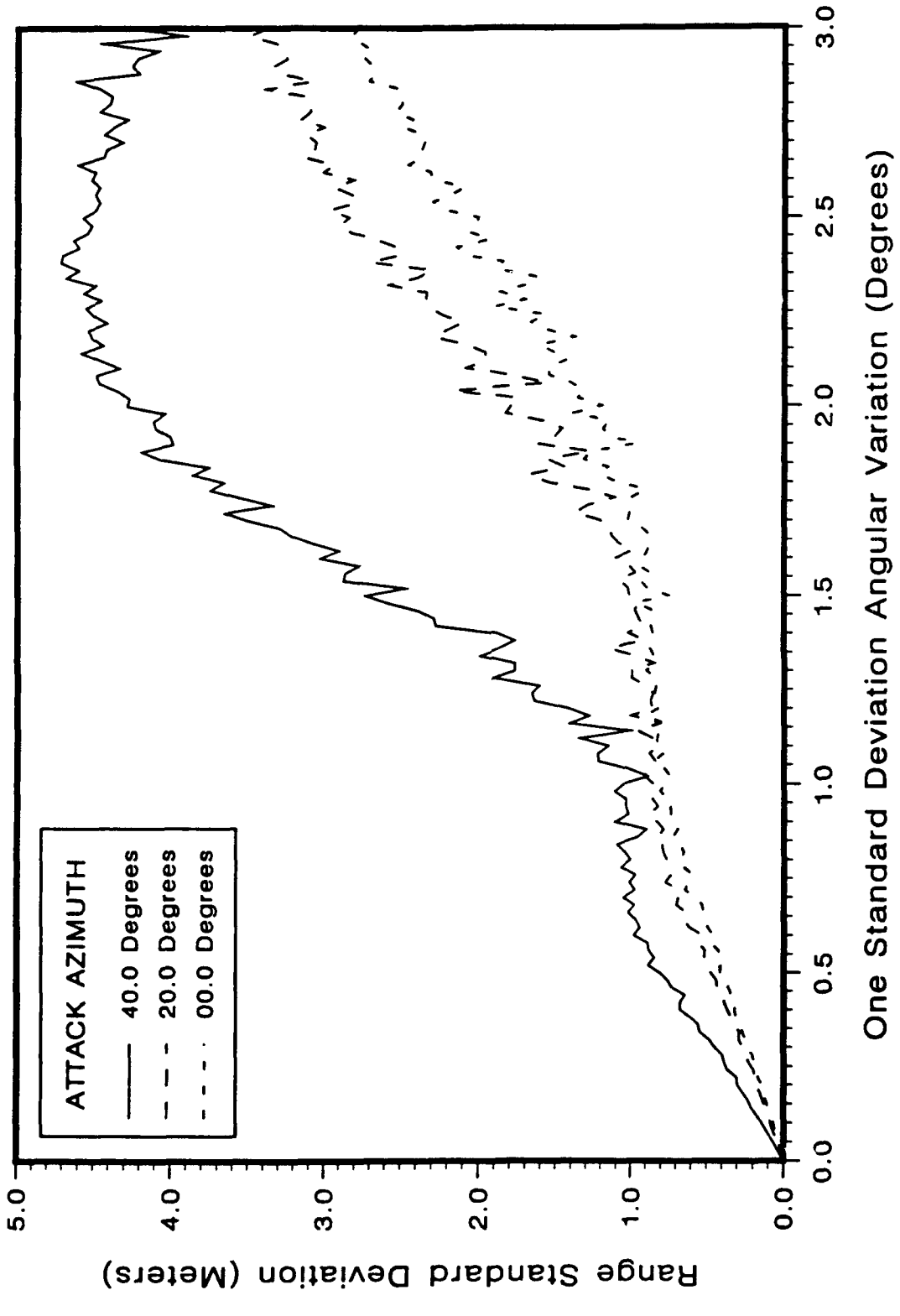

Figure 15. Range Standard Deviation for LOB Tracker Simulation with Angular Variations from 0.0 to 3.0 Degrees

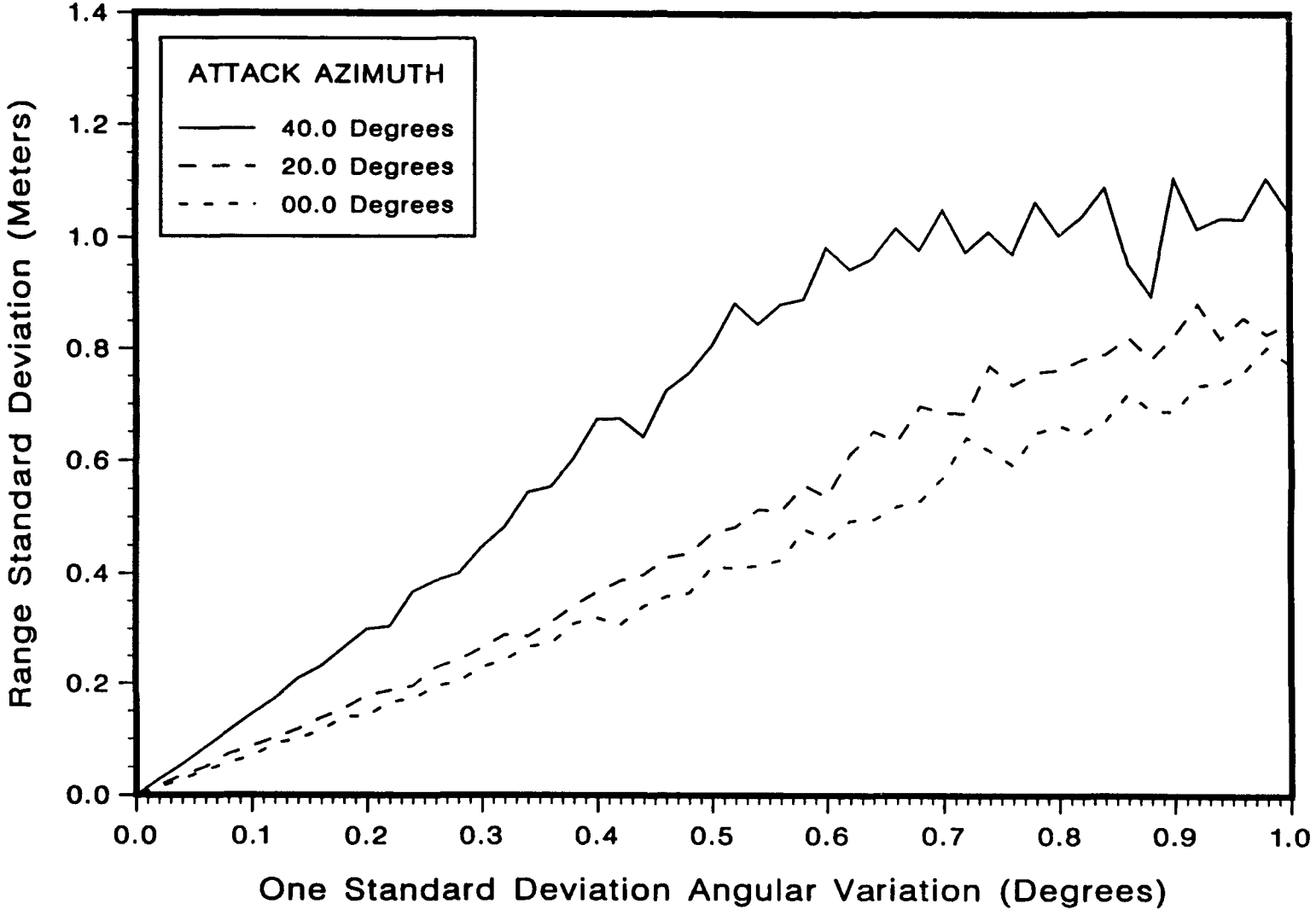

Figure 16. Range Standard Deviation for LOB Tracker Simulation with Angular Variations from 0.0 to 1.0 Degrees

For each flight, observations were taken every 0.0005 s (0.15 m) along the projectile's straight line trajectory until it's estimated position crosses the 11-m decision line (y = 11.0), at which time the estimated time-to-intercept was compared with the actual time-to-intercept (computed in a separate routine) and the difference recorded. During a flight, any observed projectile location that was at a y distance equal to or less than 0 or greater than 40 m was discarded. Also, any estimated impact location (on the sensor base line) which was more than 20 m to the right of the right sensor or more than 20 m to the left of the left sensor was discarded. This process was repeated for 500 flights using different random streams to generate Gaussian LOB errors with the desired standard deviations. After the completion of 500 iterations, the time-to-intercept differences were combined to compute their mean and standard deviation. These means and standard deviations, expressed in seconds, were multiplied by the true projectile velocity to convert the results to range distances. Then the specified angular standard deviation was changed and the process (another 500 flights) repeated.

Figure 13 shows the range bias (mean) in meters as a function of sensor angular variations from 0°–3° (one standard deviation) for three attack azimuths (0°, 20°, and 40°). It should be noted that an attack azimuth of 0° is a head-on (or on-boresight) attack, a 20° attack azimuth comes from the direction 20° left of center, etc. For attack azimuths of 0° and 20°, the range biases are similar and settle to values between ±1.0 m for angular variations of ±2.2° or less. For the 40° attack azimuth, the range biases are much larger and don't settle between ±1.0 m until the angular variations are less than ±1.5°. A positive range bias is indicated by an average location estimate that is closer to the intercept line than the true projectile location and a negative range bias value indicates a location estimate that is further from the intercept line than the true projectile location. Overestimated projectile speeds and/or underestimated projectile ranges can cause the range bias to be positive. Underestimates of projectile speeds and/or overestimates of projectile ranges can cause the range bias to be negative. In addition, the discarding of unacceptable locations, i.e. those outside the input cutoff ranges, affects the range bias.

Figure 14 is a plot of the same data used in Figure 13 with different scales for the axes. In Figure 14 it will be noted that for angular variations of below 0.5°, the range bias is between ±0.1 m for all attack azimuths.

Figure 15 is a plot of the range standard deviations (e.g., the computed standard deviation about the means plotted in Figure 13). In Figure 15, the standard deviations for all attack azimuths exhibit some random variations. The standard deviations for the 0° and 20° attack

azimuths tend to settle down at angular variations less that 1.1°, while for the 40° attack azimuth, it settles down below 0.7° angular variation.

Figure 16 is a plot of the same data used in Figure 15 with different scales for the axes. Here, for an angular variation of below 0.33°, the range standard deviation are less than 0.5 m for each of the attack azimuths. Using Figures 14 and 16, it can be determined that for angular variations of 0.33° or less, the estimated range bias would be 0.4 ±0.5 m for the 40° attack azimuth and less for the 0° and 20° attack azimuths.

6.5 Closed-Form Analysis of the Scenario. A lower bound for range errors of the system described in the scenario (Section 6.3) was found by using equation 13 from Appendix G. The estimates of the range variance were found using the subroutine LOBCALC from LOBCA of Appendix B. These lower bounds are presented in Figures 17 and 18 and are superimposed on Figures 15 and 16. Note these lower bounds are upper bounds of system performance.

A comparison of the lower bound with the simulation results shows that simulation performance starts to track the lower bound when the LOB errors fall below 0.7. It was observed that the variance estimates of the simulation don't cross the theoretic lower bound after the errors are below 0.7. This indicates that the variability in estimating the variance has diminished to an acceptable level. The results became acceptable whenever the determinant of the trajectory parameters (slope and intercept) variance matrix was 0.005 or greater. The instability of the system indicated by a small determinant can be reduced by increasing the detection range or increasing the sampling rate. Below 0.7 LOB error, the simulation results are about 10% higher than the lower bound indicating that there is room for improvement. This could be partially achieved by using a weighted least-squares estimator or perhaps by using Sequential Regression (SER) techniques. The simplicity of the SER estimators make them a more desirable choice than a technique based on weighted regression. It may prove difficult to find a good method to set the fading parameter in an SER estimator. Different methods for setting weights for weighted least squares would need to be tested and compared through the use of simulations. If the errors associated with the observations are correlated, the complexity of the estimator may need to be increased; again, ideas need to be tested through simulations. It is also possible that the simple and fast estimation procedure in the simulation is good enough and there is no merit in an increase of system complexity.

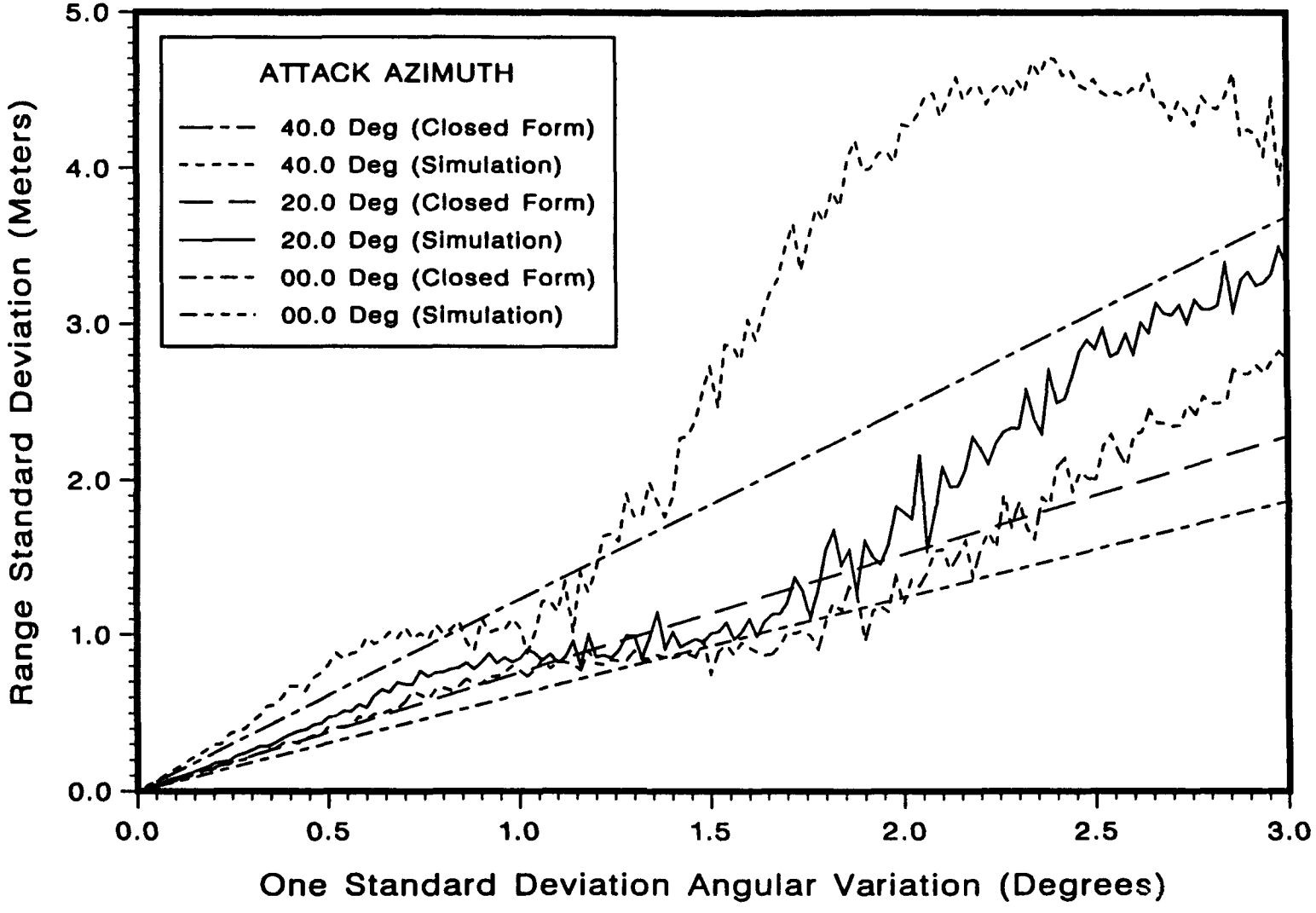

Figure 17. Range Standard Deviation, Closed Form vs Simulation, for LOB Tracker with Angular Variations from 0.0 to 3.0 Degrees

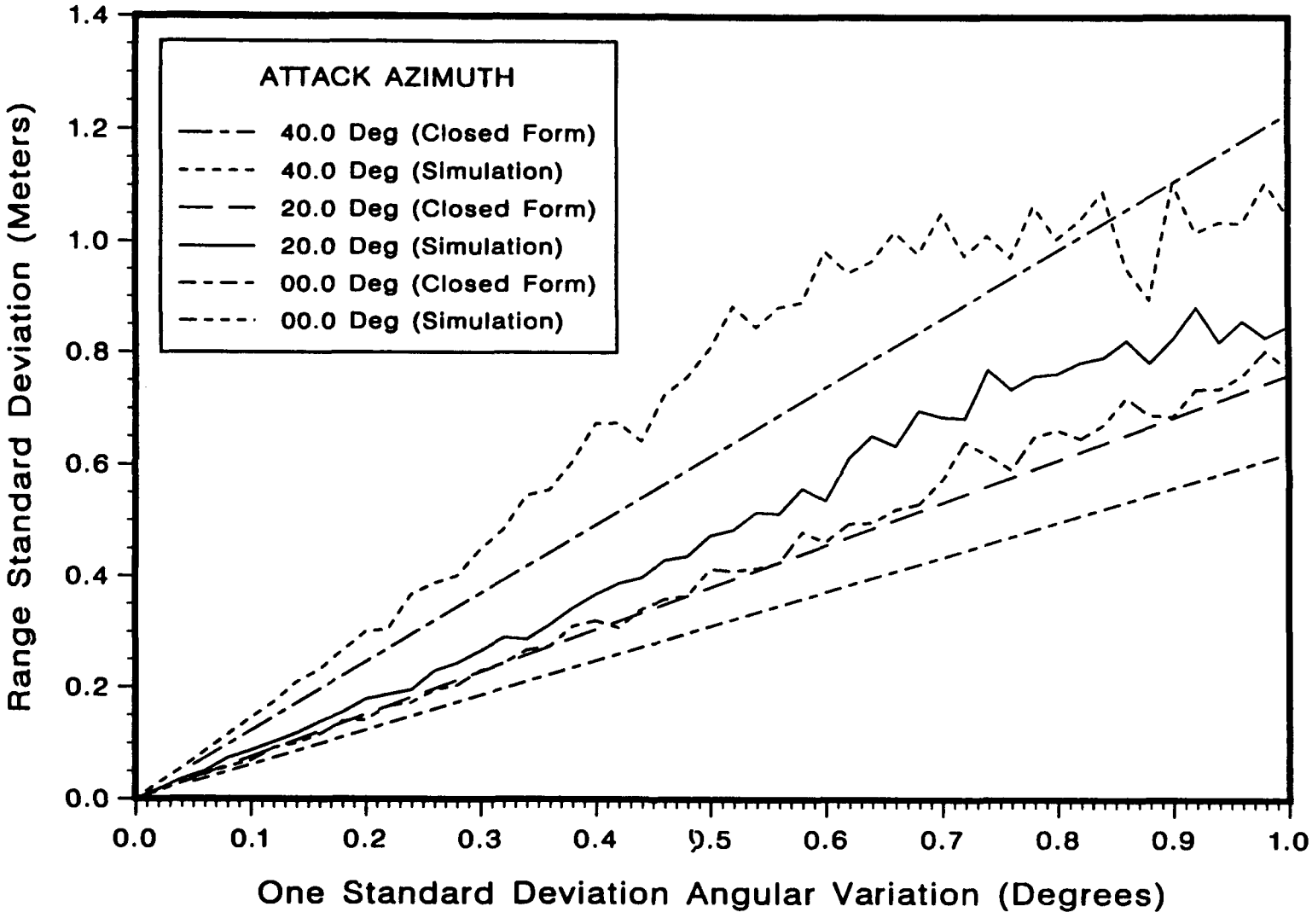

Figure 18. Range Standard Deviation, Closed Form vs Simulation, for LOB Tracker with Angular Variations from 0.0 to 1.0 Degrees

7. FINAL COMMENTS

Every analysis of a system that uses LOB information concerns itself with the transformation of the errors as they are introduced and then propagate through the subsystems. This report has focused on the use of LOB data, rather than the specific error transformations that go into the creation of a specific LOB measurement. If specific engineering models are available, they can be used a priori to determine the error to associate with specific measurements. Both simulation and closed-form methods are valuable tools in determining expected and potential system performance.

Parametric studies can be used to map the system's response surface. For example, in the scenario of Section 6.3, it may be of interest to map system performance over initial detection range, sampling rate, and LOB error. Also, it may be of interest to compare system simulation results with the determinant of the parameters covariance matrix. Another possible study is to investigate changing the rules for rejecting data. The simulation can also be used to quantify the effects of correlated errors on the estimator. The information from a parametric study could then be used to impact system design and indicate the most economical approach before hardware is developed or acquired.

It is possible to extend the ideas herein to consider three dimensions and to use other types of information in conjunction with LOB information. Alexander (1980, 1981) discusses the use of time difference of arrival, differential Doppler, and LOB information for locating stationary emitters. Thompson (1991b) discusses using range and LOB information to locate targets in both two and three dimensions.

The appropriateness of using the bivariate normal distribution as a model for the errors should be investigated. From the previous discussion of extreme range values it can be stated that the errors are not exactly distributed as a bivariate normal. A study could be performed to asses the conditions under which this assumption is reasonable.

This report includes and references many ideas and tools used to analyze systems that use LOB information. Hopefully this presentation will help the reader understand and evaluate LOB systems.

8. REFERENCES

Alexander, C. "Techniques to Evaluate the Performance of Tactical Airborne Geolocation Systems." NSA-TR-R56/01/80, National Security Agency, 1980.

Alexander, C. "Additional Techniques to Evaluate the Performance of Tactical Airborne Geolocation Systems." NSA-TR-R56/02/81, National Security Agency, 1981.

Alexander, C. "Geolocation Performance of Spread Spectrum Emitters Using TDOA/DD/AOA Processing." NSA-TR-E51-001-91, National Security Agency, 1991.

Bjerhammar, A. Theory of Errors and Generalized Matrix Inverses. Elsevier Scientific Publishing, 1973.

Bunts, P. W. "User's Manual for the Geolocation Error Model." IBM Technical Report, 1982.

Daniels, H. E. "The Theory of Position Finding." Journal of the Royal Statistical Society, 1951.

Gelb, A. Editor, Applied Optimal Estimation, M.I.T. Press, 1974.

Grubbs, F. "Statistical Measures of Accuracy for Riflemen and Missile Engineers." 4109 Webster Rd., Havre de Grace, MD, 1964.

Hutton, M. F. "GEM Interferometer Cochannel Interference Model." IBM Technical Report, 1984.

Hutton, M. F., and C. Alexander. "Relative Accuracy of Interferometer and Amplitude Comparison DF Systems." NSA-TR-5200-R506, National Security Agency, 1984.

Koopman, B. O. Search and Screening: General Principles with Historic Applications. Peragamon Press, 1980.

Kumar, P. R., and P. Varayia. Stochastic Systems Estimation, Identification and Adaptive Control.

Law, A. M., W. D. Kelton. Simulation Modeling and Analysis. McGraw Hill, 1982.

Maybeck, P. S. Stochastic Models, Estimation and Control, Vol. 1, Academic Press, 1979.

Skolnik, M. I. Introduction to Radar Systems. McGraw Hill, 1980.

Spriet, J. A., and G. C. Vansteenkiste. Computer Aided Simulation Modeling, Academic Press, 1982.

Stratton, S. R., H. B. Wallace, and D. G. Bauerle. "Multipath Induced Tracking Errors at 95 and 140 GHz." BRL-MR-3947, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1991.

Thompson, A. A. "Data Fusion for Least Squares." BRL-TR-3303, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1991a.

Thompson, A. A. "A Model of a Range-Angle of Arrival Sensor System for an Active Protection System." BRL-TR-3243, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1991b.

Thompson, A. A. "Information Reduction Due to Correlation." BRL-TR-3331, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1992.

Torrieri, D. J. "Statistical Theory of Passive Location Systems." IEEE Transactions on Aerospace and Electronic Systems, vol. 20, No. 2, March 1984.

Wallace, H. B. "140-GHz Capture Antenna Multipath Experiment." ARBRL-MR-02855, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1978.

Wegner, L. H. "On the Accuracy Analysis of Airborne Techniques for Passively Locating Electromagnetic Emitters." R-722-PR, Rand Corporation, 1971.

Widrow, B., and S. D. Stearns. Adaptive Signal Processing, Prentice-Hall, 1985.

Wiley, R. G. Electronic Intelligence: The Interception of Radar Signals. Artec House, 1985.

Wilson, C. L., "Bearing Angle Errors in Interferometer Antenna Systems Used for Short Range Verticle Angle Measurements." BRL-TR-1701, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1968.

APPENDIX A:

NATIONAL SECURITY AGENCY (NSA) MEMORANDUM FOR RECORD,
DR. CHARLES ALEXANDER

INTENTIONALLY LEFT BLANK.

SUBJECT: ERROR ELLIPSE IN DOA MEASUREMENTS

This memorandum describes the errors that are incurred when an emitter is located by direction-of-arrival measurements. It will be assumed that the direction of arrival of the signal is measured at only two points and that the resulting directions of arrival lie in the same plane. If x and y are Cartesian coordinates in this plane, the geometry will be as shown in Figure 1.

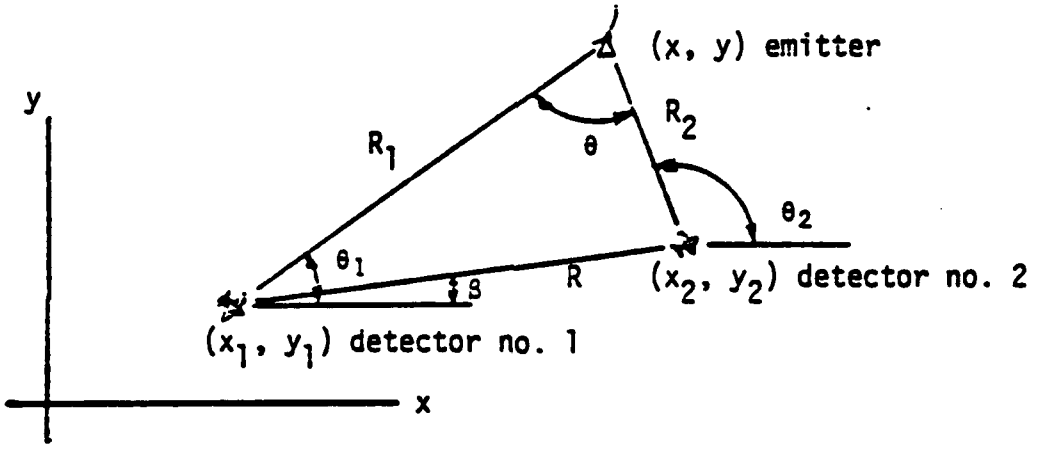

Figure 1. Geometry of a DOA System

Here the emitter located at (x, y) is detected by two detectors at (x_1, y_1) and (x_2, y_2), and the angles of arrival of the signal at these two detectors are θ_1 and θ_2, measured from an arbitrary direction which is taken to be the x-axis. The separation of the detectors is R, the bearing angle of one from the other is β, their distances from the emitter are R_1 and R_2, and the angle they subtend at the emitter is θ. Errors $\Delta\theta_1$ and $\Delta\theta_2$ in the

measured values of θ_1 and θ_2 then give rise to errors in the measured position of the emitter. It will be assumed that the measurements of θ_1 and θ_2 are unbiased, so that the measured values of these angles are the expected values of their statistical distributions. Furthermore, the errors $\Delta\theta_1$ and $\Delta\theta_2$ will be assumed to be uncorrelated and to have standard deviations σ_1 and σ_2 respectively. Denoting average values by a bar, these assumptions are

$$\begin{aligned} \overline{\Delta\theta_1} &= \overline{\Delta\theta_2} = 0 \\ \overline{\Delta\theta_1 \Delta\theta_2} &= 0 \\ \overline{(\Delta\theta_1)^2} &= \sigma_1^2 \\ \overline{(\Delta\theta_2)^2} &= \sigma_2^2 \end{aligned} \tag{1}$$

It will be further assumed that the errors in determining the coordinates of the two detectors can be neglected.

When x_1, x_2, y_1, y_2, and the measured values of θ_1 and θ_2 are known, the coordinates (x, y) of the emitter are determined by the equations:

$$\begin{aligned} \tan\theta_1 &= \frac{y-y_1}{x-x_1} \\ \tan\theta_2 &= \frac{y-y_2}{x-x_2} \end{aligned} \tag{2}$$

from which it is found that

$$x = \frac{x_1 \tan \theta_1 - x_2 \tan \theta_2 - y_1 + y_2}{\tan \theta_1 - \tan \theta_2}$$

$$y = \frac{(x_1-x_2) \tan \theta_1 \tan \theta_2 + y_2 \tan \theta_1 - y_1 \tan \theta_2}{\tan \theta_1 - \tan \theta_2} \qquad (3)$$

If it is assumed that the system is capable of determining the location of the emitter with reasonable accuracy, then the errors in the measurement of x and y will be much smaller than the distance between the detectors or the distance of the emitter from either one of the detectors. The position errors Δx and Δy produced by measurement errors $\Delta\theta_1$ and $\Delta\theta_2$ can then be found by differentiation of Eqs. 3 with respect to θ_1 and θ_2, and they are

$$\Delta x = A \, \Delta\theta_1 + B \, \Delta\theta_2$$

$$\Delta y = A \tan \theta_2 \, \Delta\theta_1 + B \tan \theta_1 \, \Delta\theta_2 \qquad (4)$$

where

$$A \equiv \frac{(x_2-x_1) \tan \theta_2 - (y_2-y_1)}{\cos^2 \theta_1 \, (\tan \theta_1 - \tan \theta_2)^2}$$

$$B \equiv \frac{-(x_2-x_1) \tan \theta_1 + (y_2-y_1)}{\cos^2 \theta_2 \, (\tan \theta_1 - \tan \theta_2)^2} \qquad (5)$$

These relations can be expressed more simply by noting from Figure 1 that $\theta = \theta_2 - \theta_1$, that the vertex angles in the triangle of Figure 1 at the points (x_1, y_1) and (x_2, y_2) are $\theta_1 - \beta$ and $\pi - \theta_2 + \beta$ respectively, and that $\cos \beta = (x_2-x_1)/R$ and $\sin \beta = (y_2-y_1)/R$. It is then seen from the law of sines that

$$\sin \theta = \frac{(x_2-x_1) \sin \theta_2 - (y_2-y_1) \cos \theta_2}{R_1}$$

$$= \frac{(x_2-x_1) \sin \theta_1 - (y_2-y_1) \cos \theta_1}{R_2}$$

These equations can be written

$$R_1 = \frac{(x_2-x_1) \tan \theta_2 - (y_2-y_1)}{\cos \theta_1 (\tan \theta_2 - \tan \theta_1)}$$

$$R_2 = \frac{(x_2-x_1) \tan \theta_1 - (y_2-y_1)}{\cos \theta_2 (\tan \theta_2 - \tan \theta_1)}$$

Also, the fact that $\theta = \theta_2-\theta_1$ leads directly to

$$\sin \theta = \cos \theta_1 \cos \theta_2 (\tan \theta_2 - \tan \theta_1)$$

The coefficients A and B of Equation 5 can then be written

$$A = \frac{R_1 \cos \theta_2}{\sin \theta}$$

$$B = -\frac{R_2 \cos \theta_i}{\sin \theta}$$

and Eqs. 4 become

$$\Delta x = (R_1 \Delta\theta_1 \cos \theta_2 - R_2 \Delta\theta_2 \cos \theta_1)/\sin \theta$$

$$\Delta y = (R_1 \Delta\theta_1 \sin \theta_2 - R_2 \Delta\theta_2 \sin \theta_1)/\sin \theta \qquad (6)$$

By use of Eqs. 1, the means and variances of Δx and Δy are then easily found to be

$$\overline{\Delta x} = \overline{\Delta y} = 0$$

$$\overline{(\Delta x)^2} = (R_1^2 \sigma_1^2 \cos^2 \theta_2 + R_2^2 \sigma_2^2 \cos^2 \theta_1)/\sin^2 \theta$$

$$\overline{(\Delta y)^2} = (R_1^2 \sigma_1^2 \sin^2 \theta_2 + R_2^2 \sigma_2^2 \sin^2 \theta_1)/\sin^2 \theta \qquad (7)$$

$$\overline{\Delta x \Delta y} = (R_1^2 \sigma_1^2 \sin 2\theta_2 + R_2^2 \sigma_2^2 \sin 2\theta_1)/(2 \sin^2 \theta)$$

To visualize the measurement error, it is convenient to express it in terms of coordinates x' and y' which are rotated through an angle α relative to x and y. In these coordinates

$$\Delta x' = \Delta x \cos \alpha + \Delta y \sin \alpha$$

$$\Delta y' = -\Delta x \sin \alpha + \Delta y \cos \alpha \qquad (8)$$

and the means and variances of $\Delta x'$ and $\Delta y'$ are found to be

$$\overline{\Delta x'} = \overline{\Delta y'} = 0$$

$$\begin{aligned}
\overline{(\Delta x')^2} &= \overline{(\Delta x)^2}\cos^2\alpha + 2\overline{\Delta x \Delta y}\sin\alpha\cos\alpha + \overline{(\Delta y)^2}\sin^2\alpha \\
&= \tfrac{1}{2}\left[\overline{(\Delta x)^2} + \overline{(\Delta y)^2}\right] + \tfrac{1}{2}\left[\overline{(\Delta x)^2} - \overline{(\Delta y)^2}\right]\cos 2\alpha \\
&\quad + \overline{\Delta x \Delta y}\sin 2\alpha \\
\overline{(\Delta y')^2} &= \overline{(\Delta x)^2}\sin^2\alpha - 2\overline{\Delta x \Delta y}\sin\alpha\cos\alpha + \overline{(\Delta y)^2}\cos^2\alpha \\
&= \tfrac{1}{2}\left[\overline{(\Delta x)^2} + \overline{(\Delta y)^2}\right] - \tfrac{1}{2}\left[\overline{(\Delta x)^2} - \overline{(\Delta y)^2}\right]\cos 2\alpha \\
&\quad - \overline{\Delta x \Delta y}\sin 2\alpha \\
\overline{\Delta x' \Delta y'} &\equiv -\overline{(\Delta x)^2}\sin\alpha\cos\alpha + \overline{\Delta x \Delta y}(\cos^2\alpha - \sin^2\alpha) \\
&\quad + (\Delta y)^2\sin\alpha\cos\alpha \\
&= \tfrac{1}{2}\left[\overline{(\Delta y)^2} - \overline{(\Delta x)^2}\right]\sin 2\alpha + \overline{\Delta x \Delta y}\cos 2\alpha
\end{aligned} \tag{9}$$

Obviously the covariance $\overline{\Delta x' \Delta y'}$ vanishes if the coordinates are rotated through the angle α_o defined by

$$\tan 2\alpha_o = \frac{2\overline{\Delta x \Delta y}}{\overline{(\Delta x)^2} - \overline{(\Delta y)^2}} \tag{10}$$

When $\alpha = \alpha_o$, the variances $\overline{(\Delta x')^2}$ and $\overline{(\Delta y')^2}$ have their principal values, which are given by

$$\overline{(\Delta x')^2} = \tfrac{1}{2}\left\{ \overline{(\Delta x)^2} + \overline{(\Delta y)^2} + \sqrt{\left[\overline{(\Delta x)^2} - \overline{(\Delta y)^2}\right]^2 + 4\overline{(\Delta x \Delta y)^2}} \right.$$

$$\overline{(\Delta y')^2} = \tfrac{1}{2}\left\{ \overline{(\Delta x)^2} + \overline{(\Delta y)^2} - \sqrt{\left[\overline{(\Delta x)^2} - \overline{(\Delta y)^2}\right]^2 + 4\overline{(\Delta x \Delta y)^2}} \right. \qquad (11)$$

When $\overline{(\Delta x)^2}$, $\overline{(\Delta y)^2}$, and $\overline{\Delta x \Delta y}$ are given by Eqs. 7, Eqs. 10 and 11 can be rewritten

$$\tan 2(\alpha_o - \theta_1) = \frac{\sin 2\theta}{\frac{R_2^2\sigma_2^2}{R_1^2\sigma_1^2} + \cos 2\theta}$$

$$\overline{(\Delta x')^2} = \frac{1}{2\sin^2\theta}\left[R_1^2\sigma_1^2 + R_2^2\sigma_2^2 + \sqrt{R_1^4\sigma_1^4 + R_2^4\sigma_2^4 + 2R_1^2\sigma_1^2 R_2^2\sigma_2^2 \cos 2\theta}\right] \qquad (12)$$

$$\overline{(\Delta y')^2} = \frac{1}{2\sin^2\theta}\left[R_1^2\sigma_1^2 + R_2^2\sigma_2^2 - \sqrt{R_1^4\sigma_1^4 + R_2^4\sigma_2^4 + 2R_1^2\sigma_1^2 R_2^2\sigma_2^2 \cos 2\theta}\right]$$

These equations determine the orientation of the principal axes and the variances of the errors in the direction of these axes in terms of the errors $R_1\sigma_1$ and $R_2\sigma_2$ and the angle of intersection θ. The variance of the component of error $\Delta x'$ in an arbitrary direction α can be found from the first of Eqs. 9 when $\overline{(\Delta x)^2}$, $\overline{(\Delta y)^2}$, and $\overline{\Delta x \Delta y}$ are given by Eqs. 7, and it is

$$\overline{(\Delta x')^2} = \frac{1}{2\sin^2\theta}\left[R_1^2\sigma_1^2 + R_2^2\sigma_2^2 + \sqrt{R_1^4\sigma_1^4 + R_2^4\sigma_2^4 + 2R_1^2 R_2^2\sigma_1^2\sigma_2^2 \cos 2\theta}\right. \qquad (13)$$

$$\left. \times \cos 2(\alpha - \alpha_o)\right]$$

where α_o is given by the first of Eqs. 12.

In the particular case in which $\Delta\theta_1$ and $\Delta\theta_2$ are normally distributed, $\Delta x'$ and $\Delta y'$ will also be normally distributed, and they will be independent if the coordinates x', y' are rotated to the angle $\alpha = \alpha_o$. The probability density of the location of the emitter will then be constant at all points on an ellipse whose axes lie along these coordinate axes and have lengths proportional to $\sqrt{\overline{(\Delta x')^2}}$ and $\sqrt{\overline{(\Delta y')^2}}$. When the size of the ellipse is such that the probability that it contains the emitter is 1/2, the ellipse has been called the probable ellipse associated with the location of the emitter. The semi-axes of this ellipse are $\sqrt{\ell n 4}$ or 1.1774 times $\sqrt{\overline{(\Delta x')^2}}$ and $\sqrt{\overline{(\Delta y')^2}}$.

APPENDIX B:

LOBCA CODE

INTENTIONALLY LEFT BLANK.

User's Guide for LOBCA

The program LOBCA calculates the error covariance for a LOB system. For online help simply type a question mark following the program name. If the program name is typed the program runs with a set of default values for each of the program parameters. These values can be modified by setting the values of specific flags that follow the program as arguments. The following describes the flags and gives several examples of running the program. It is assumed that the executable version of the program is called lobca.

-b This is the flag for the offboresight angle. The default value is 0. To change it to 45 degrees the flag -b should be followed by 45. All flags can be separated by a space if desired. Example lobca -b45 or lobca -b 45

-e This flag sets the starting value for the lob error. Default is 1.

-f This flag sets the ending value of the lob error. Default is 3.

-g This flag sets the value of the lob stepsize. Default is 1.

-r This flag sets the beginning range to the target. Default is 1.

-s This flag sets the ending range of the target. Default is 3.

-t This sets the range stepsize. the default is 1 .

Example
lobca -b20 -e.1 -f.8 -g.15 -r2 -s5 -g.5

Appendix B

```
#include <math.h>
void lobcalc(float x1,float y1,float x2,float y2,float xt, float yt,float sda1,float sda2);
void main(int argc, char *argv[]);

void main(int argc, char *argv[])
/* This is a driver program for the line of bearing program that follows */
{
 float tr,x1,y1,x2,y2,xt,yt,sda1,sda2,step;
 float tr_b,tr_e,tr_step,err_b,err_e,err_step,aoa,x_fac,y_fac;
 int i,j;
 char c,argument[30],option;

 i=1;
 x1=0;
 y1=.5;
 x2=0;
 y2=-.5;
 xt=0;
 yt=0;
 tr_b=1;
 tr_e=3;
 tr_step=1;
 aoa=0;
 err_b=1;
 err_e=3;
 err_step=1;

 while (i<argc)
 {
  j=strlen(argv[i]);
  c=argv[i][0];
  switch (c)
  {
   case '-' :option=argv[i][1];
          if (j>2)
            strcpy(argument,&(argv[i][2]));
           else
            {
            i++;
            strcpy(argument,argv[i]);
            }
          switch (option)
          {
```

```
case 'b': if ((aoa=atof(argument))>180)
      {
       printf("0ff Boresight Angle too large");
       printf("0rogram stopped");
       exit(1);
      }
      i++;
      break;
case 'r': if ((tr_b=atof(argument))<0)
       {
        printf("0arget Range Start must be positive");
        exit(1);
       }
       i++;
       break;
case 's': if ((tr_e=atof(argument))<0)
       {
        printf("0arget Range Endpoint must be positive");
        exit(1);
       }
       i++;
       break;
case 't': if ((tr_step=atof(argument))<0)
       {
        printf("0arget Range STEP must be positive");
        exit(1);
       }
       i++;
       break;
case 'e': if ((err_b=atof(argument))<0)
       {
        printf("0OB error BEGIN must be positive");
        exit(1);
       }
       i++;
       break;
case 'f': if ((err_e=atof(argument))<0)
       {
        printf("0OB error Final must be positive");
        exit(1);
       }
       i++;
       break;
case 'g': if ((err_step=atof(argument))<0)
       {
        printf("0arget Range STEP must be positive");
```

```
            exit(1);
            }
            i++;
            break;
        default : printf("0nknown option request");
            strcpy(argv[i],"mistake");

        }
        break;
  default :printf("0mproper option for argument");
        printf(" options are -b off boresight angle default is 0");
        printf("          -e start value for lob error default=1");
        printf("          -f end value of lob error default = 3");
        printf("          -g step size for lob error default=1");
        printf("          -r begining range to target in baselines default = 1");
        printf("          -s end value of target range default=3baselines");
        printf("          -t range step default = 1");
        exit(1);
 }

}
aoa=aoa/180*M_PI;
x_fac=cos(aoa);
y_fac=sin(aoa);
for (tr=tr_b; tr<tr_e; tr+=tr_step)
{
 xt=tr*x_fac;
 yt=tr*y_fac;
 for ( step=err_b; step < err_e; step+=err_step)
 {
  sda1=M_PI/180*step;
  sda2=sda1;
  lobcalc(x1,y1,x2,y2,xt,yt,sda1,sda2);
 }
}
}


void lobcalc(float x1,float y1,float x2,float y2,
        float xt,float yt,float sda1,float sda2)
/* This program is based the memoradum for record written by
   Dr. Charles Alexander of NSA it is a standard method for finding
   the errors of any line of bearing system.              */

{
 float sin1,sin1sq,sin2,sin2sq,cos1,cos1sq,cos2,cos2sq;
```

```
float r1,r2,r,sine,sinesq;
float v1,v2,v3,v4,xvar,yvar,sdx,sdy,vara1,vara2;
float sin2a1,sin2a2,r1sq,r2sq,varxy,axis1,axis2;

printf("0*****************************************************************");
printf("0ensor 1 at %.2f , %.2f ",x1,y1);
printf("0ensor 2 at (%.2f , %.2f)",x2,y2);
printf("0arget is located at (%.2f,%.2f)",xt,yt);
v1=57.29578*sda1;
printf("0rror of sensor 1 is %.3f radians or %.3f degrees",sda1,v1);
v1=57.29578*sda2;
printf("0rror of sensor 2 is %.3f radians or %.3f degrees",sda2,v1);

v1=x1-xt;
v2=y1-yt;
r1sq=v1*v1+v2*v2;
r1=sqrt( r1sq );

v3=x2-xt;
v4=y2-yt;
r2sq=v3*v3+v4*v4;
r2=sqrt(r2sq);

sin1=-v2/r1;
cos1=-v1/r1;
sin2=-v4/r2;
cos2=-v3/r2;
sin1sq=sin1*sin1;
cos1sq=cos1*cos1;
sin2sq=sin2*sin2;
cos2sq=cos2*cos2;

sine=((x1-x2)*sin2-(y1-y2)*cos2)/r1;
sinesq=sine*sine;

vara1=sda1*sda1;
vara2=sda2*sda2;

xvar=(r1sq*vara1*cos2sq + r2sq*vara2*cos1sq)/sinesq;
yvar=(r1sq*vara1*sin2sq + r2sq*vara2*sin1sq)/sinesq;
v1=asin(sin1);
v2=asin(sin2);
sin2a1=sin(2*v1);
sin2a2=sin(2*v2);
varxy=(r1sq*vara1*sin2a2 + r2sq*vara2*sin2a1)/(2*sinesq);
```

```
printf("0he X standard deviation was : %f",sqrt(xvar));
printf("0he Y standard deviation was : %f ",sqrt(yvar));

v1=xvar-yvar;
v3=sqrt(v1*v1+4*varxy*varxy);
v2=xvar+yvar;
axis1=(v2+v3)/2;
axis2=(v2-v3)/2;

printf("0he major axis standard deviation is %f",sqrt(axis1));
printf("0he minor axis standard deviation is %f",sqrt(axis2));

}
```

APPENDIX C:

RANLOB CODE

INTENTIONALLY LEFT BLANK.

Users Guide for RANLOB

This program represents two observers using line of bearing measurements to locate a target. The simulation is confined to the XY plane; variations due to a third dimension are ignored. Self location errors of the observers and angle measurement errors by the observers are considered.

Before running this program a data file must be create and associated with fortran logical file 4. This is done by the command

```
assign datfil.dat for004
```

where datfil.dat is the name of the data file to be used by the program. Following is a line by line desciption of the five line input file. Note that by changing the 4s to 5s the input would become interactive.

LINE 1

Two real numbers representing the location of observer 1.

LINE 2

Two real numbers representing the location of observer 2.

LINE 3

Two real numbers representing the location of the target. This X value must be larger than either of the observer X values. The program uses an ARCTAN function and assumes that the target has a larger X coordinate than either of the observers.

LINE 4

A real number that represents the error in angle measurement of both observers.

LINE 5

Two real numbers representing the errors in self location made by the observers in the x and y dimensions.

The ouput report gives several statistical measurements of performance. These are based on a sample size of twenty. The target location and the predicted target location are given. The standard deviation of the X and Y coordinates is given. The radial standard deviation and CEP are also reported.

The program is located on AMSAA VMS in [athompson.lob]. to run the program type

```
run [athompson.lob]ranlob
```

Code of RANLOB

```
c this program generates intersections
c of two lines based on the addition of errors to
c the locations of the two referance points and
c errors in measuring an angle to the intersection point
c
c the intended use is to mimic line of bearing direction
c finding systems
c
c errors are generated based on a gaussian distribution
c the input data should be assigned to for004
c    assign <filename> for004
c format of the input file should be
c location of sensor1 x and y (two numbers)
```

```
c location of sensor2 x and y (two numbers)
c location of target x and y (two numbers)
c standard deviation of lob measurement in degrees (one number)
c standard deviation of x an y measurements (twonumbers)
c
c due to the use of the arctan function the sensors should be on the left
c
c Andrew Thompson
      implicit real (a-z)
      dimension xl(100),yl(100)
      integer iseed,i,n
      n=20
      iseed=3
c platforms on the left
c target ot the right
      read(4,*)xp1,yp1
      read(4,*)xp2,yp2
      read(4,*)xt,yt
      read(4,*)sdlob
      read(4,*)sdx,sdy
c
c find the actual angles and tangents
c
      tanp1t=(yt-yp1)/(xt-xp1)
      tanp2t=(yt-yp2)/(xt-xp2)
      angp1t=atand(tanp1t)
      angp2t=atand(tanp2t)
c
c start simulation
c
      do i=1,n
         call gauss(iseed,g1,g2)
         ang1=angp1t+g1*sdlob
         ang2=angp2t+g2*sdlob
         m1=tand(ang1)
         m2=tand(ang2)
         call gauss(iseed,g1,g2)
         x1=xp1+g1*sdx
         y1=yp1+g2*sdy
         call gauss(iseed,g1,g2)
         x2=xp2+g1*sdx
         y2=yp2+g2*sdy
         b1=y1-m1*x1
         b2=y2-x2*m2
         x=(b2-b1)/(m1-m2)
         y=m1*x+b1
         write(6,*)x,y
         xl(i)=x
         yl(i)=y
       enddo
c
c go calculate statistics
c
       call stat(xl,yl,n,xt,yt)
       end
c
c ********************************************************
c function to produce two gaussian variables g1,g2
c ********************************************************
c
      subroutine gauss(iseed,g1,g2)
    5 u=2*ran(iseed)-1
      v=2*ran(iseed)-1
      r=u*u+v*v
      if (r.ge.1) goto 5
      s=sqrt(-2*alog(r)/r)
```

```
      g1=u*s
      g2=v*s
      end
c Andrew Thompson
c*****************************************************************
c ****************************************************************
      subroutine stat(xl,yl,n,xe,ye)
      implicit real (a-z)
      integer n,nn,i
      dimension xl(n),yl(n)
      nn=n
      fval=2.51
 2000  do 10 i=1,nn
          xl(i)=xl(i)-xe
          yl(i)=yl(i)-ye
          write(6,*)xl(i),yl(i)
  10   continue
c
c
c ****************************************************************
c    statistical calculations
c ****************************************************************
      smx=0
      smy=0
      sqy=0
      sqx=0
      dn=0
      n=nn
c
      do 30 i=1,nn
        smx=smx+xl(i)
        smy=smy+yl(i)
        sqx=sqx+xl(i)*xl(i)
        sqy=sqy+yl(i)*yl(i)
  30  continue
      sxsq=(sqx-smx*smx/n)/(n-1)
      sysq=(sqy-smy*smy/n)/(n-1)
      sx=sqrt(sxsq)
      sy=sqrt(sysq)
      xbar=smx/n
      ybar=smy/n
      sxysq=sxsq+sysq
      sr=sqrt(sxysq)
c CEP computation via Grubb
      if (sxsq.gt.sysq) then
          f=sxsq/sysq
        else
          f=sysq/sxsq
      endif
      if (f.gt.fval) then
          v=2*((sxsq*sxsq+sysq*sysq)/(sxysq*sxysq))
          cep=sr*(1-v/9)**1.5
        else
          cep=1.1774*sr
      endif
c
c ****************************************************************
c output section
c ****************************************************************
      write(6,123)
  123 format(2x,'*********************************************')
      write(6,206)xe,ye
  206 format(5x,'The emitter was located at (',f9.1,',',f9.1,')')
      write(6,101)xbar,ybar
  101 format(5x,'The C of I  relative to the target was',2f13.3)
      write(6,102)sx
```

```
102 format(5x,'The standard deviation of the x component was',f14.4)
    write(6,103)sy
103 format(5x,'The standard deviation of the y component was ',f14.4)
    write(6,104)sr
104 format(5x,'The radial standard deviation was ',f14.4)
    write(6,105)cep
105 format(5x,'CEP has a radius of ',f14.4,2x,'around the C of I')
    dis=sqrt(xbar**2+ybar**2)
    if (dis.lt.cep) then
        write(6,106)
      else
        write(6,107)
    endif
106 format(5x,'The target was within the CEP')
107 format(5x,'The target was not in the CEP')
    write(6,123)
    write(6,123)
    write(6,123)
    return
    end
```

APPENDIX D:

AZEL CODE

INTENTIONALLY LEFT BLANK.

Users Guide for AZEL

This program simulates the performance of a single observer taking azimuth and elevation measurements to determine the location of a target. The errors associated with the observer are positional errors and angular measurement errors.

Before running this program an input file must be prepared. This file then must be associated with fortran unit 4. If you are on a VMS system this is accomplished by the following command.

```
ASSIGN INFIL.DAT FOR004
```

INFIL.DAT is the name of the input file. This program uses the ARCTAN function and assumes that the observer will have a lower X axis coordinate than the target. A line by line description of the five lines of input data follow. Some of the input data is not used by this program; it is there because the author felt it would be used in modifications of this program.

LINE 1

Three real numbers represents the (x,y,z) of the observer.

LINE 2

Three real numbers representing the (x,y,z) of the target. Note the z value is not used. The target is assumed to be on the ground.

LINE 3

Two real numbers representing the azimuth measurement error and the elevation measurement error. Both are assumed to be in degrees.

LINE 4

Two real numbers not used in this version. The intended use was to take into account variations in elevation of the target caused by incomplete knowledge of the ground contores.

LINE 5

The self location errors of the observer. The standard deviations associated with the X, Y, and Z coordinates of the observer. Three real numbers.

A report summarizing the performance of the system is displayed on the screen. The report includes several statistical measures of performance. These statistics are based on a sample size of twenty. The actual emitter location is printed followed by the center of impact or average for the sample. As measures of performance the standard deviations of the X and Y components are printed, the radial standard deviation is printed and the CEP is printed.

On AMSAA VMS this programis located in [athompson.stat.pk]. To run it type run [athompson.stat.pk]azel .

AZEL Code

```
c AZEL
c this program is supposed to simulate the performance of an azel location
c system (azimuth-elezation)
c theprogam works with the arctan function so the platform should be
c to the left (lower x-value) of the target.
c output will be the predicted locations
c the predicted locations relative to the actual location
c some statistics to summarize the data
c the input file will be read on for004 to link the data file use the assign
c command as follows before running the program
c              assign <filename> for004
c the input file should contain the following as indicated
c the position of the platform x,y,z    (three numbers)
c the position of the platform x,y,0    (three numbers) 0=assumed location
c the azimuth and elevation measurement errrors      (two numbers)
c the minz and maxz (two numbers) not used by this program
c the standard deviation of x,y,z    (three numbers)
c Andrew Thompson
      implicit real (a-z)
      common /list1/ zratio,reftanaz,reftanel,dist
      common /vars/ xl(100),yl(100)
      integer iseed,i,n,nn
      n=20
      nn=n
      iseed=59999999
c
c read input data
c
      read (4,*)xp,yp,zp
      read (4,*)xt,yt,zt
      read (4,*)azer,eler
      read (4,*)minz,maxz
      read (4,*)sdx,sdy,sdz
c
c calculate actual angles and the reference values of the tangents
c these reference values and the xy distance are used to find
c the location later
c
      dist=((xt-xp)**2+(yp-yt)**2)**.5
      reftanel=((zt-zp)/dist)
      elangle=atand(reftanel)
      reftanaz=((yt-yp)/(xt-xp))
      azangle=atand(reftanaz)
c
c start the simulation
c
      do i=1,nn
       call gauss(iseed,az,el)
       azgauss=az*azer
       elgauss=el*eler
       nwazang=azangle+azgauss
       nwelang=elangle+elgauss
       call gauss(iseed,dz,dd)
       zgauss=dz*sdz
       zratio=(zgauss+zp)/zp
c
c go find the value of the target in reference to the real location of the
c target
c
       call findxy(nwelang,nwazang,x,y)
c
```

```
c
c and in the errors associated with the navigational system
c note the z effects are already incorporated
c
        call gauss(iseed,dx,dy)
        xgauss=dx*sdx
        ygauss=dy*sdy
        x=x+xgauss
        y=y+ygauss
        xl(i)=x
        yl(i)=y
        write(6,*)xl(i),yl(i)
       enddo
       write(6,100)
100    format (5x,'***************    **********************')
       call stat(n,xt,yt)
       end

       subroutine findxy(nwelang,nwazang,x,y)
c
c this routine finds the location point based on the xy range and the
c the azimuth angle
c the xy range number is modified based on the different angle by
c the term ratio and is modified to acount for the incorrect z value
c by the term zratio
c
       implicit real (a-z)
       common /list1/ zratio,reftanaz,reftanel,dist
       nweltan=tand(nwelang)
       ratio=reftanel/nweltan
       ndist=ratio*dist*zratio
       x=ndist*cosd(nwazang)
       y=ndist*sind(nwazang)
       return
       end
c
c ***********************************************************
c function to produce two gaussian variables g1,g2
c ***********************************************************
c
      subroutine gauss(iseed,g1,g2)
    5 u=2*ran(iseed)-1
      v=2*ran(iseed)-1
      r=u*u+v*v
      if (r.ge.1) goto 5
      s=sqrt(-2*alog(r)/r)
      g1=u*s
      g2=v*s
      return
      end
c Andrew Thompson
c*****************************************************************
c ****************************************************************
      subroutine stat(n,xe,ye)
      implicit real (a-h),(p-z)
      common /vars/ xl(100),yl(100)
      nn=n
      fval=2.51
      do  i=1,nn
          xl(i)=xl(i)-xe
          yl(i)=yl(i)-ye
          write(6,*)xl(i),yl(i)
      enddo
c
c
c ***************************************************************
c    statistical calculations
```

```
c ********************************************************************
      smx=0
      smy=0
      sqy=0
      sqx=0
      dn=0
c
      do  i=1,nn
        smx=smx+xl(i)
        smy=smy+yl(i)
        sqx=sqx+xl(i)*xl(i)
        sqy=sqy+yl(i)*yl(i)
      enddo
      sxsq=(sqx-smx*smx/n)/(n-1)
      sysq=(sqy-smy*smy/n)/(n-1)
      sx=sqrt(sxsq)
      sy=sqrt(sysq)
      xbar=smx/n
      ybar=smy/n
      sxysq=sxsq+sysq
      sr=sqrt(sxysq)
c CEP computation via Grubb
      if (sxsq.gt.sysq) then
            f=sxsq/sysq
         else
            f=sysq/sxsq
      endif
      if (f.gt.fval) then
          v=2*((sxsq*sxsq+sysq*sysq)/(sxysq*sxysq))
          cep=sr*(1-v/9)**1.5
        else
          cep=1.1774*sr
      endif
c
c ********************************************************************************
c output section
c ********************************************************************************
      write(6,123)
  123 format(2x,'*******************************************************')
      write(6,206)xe,ye
  206 format(5x,'The emitter was located at (',f9.1,',',f9.1,')')
      write(6,101)xbar,ybar
  101 format(5x,'The C of I  relative to the target was',2f13.3)
      write(6,102)sx
  102 format(5x,'The standard deviation of the x component was',f14.4)
      write(6,103)sy
  103 format(5x,'The standard deviation of the y component was ',f14.4)
      write(6,104)sr
  104 format(5x,'The radial standard deviation was ',f14.4)
      write(6,105)cep
  105 format(5x,'CEP has a radius of ',f14.4,2x,'around the C of I')
      dis=sqrt(xbar**2+ybar**2)
      if (dis.lt.cep) then
          write(6,106)
        else
          write(6,107)
      endif
  106 format(5x,'The target was within the CEP')
  107 format(5x,'The target was not in the CEP')
      write(6,123)
      write(6,123)
      write(6,123)
      return
      end
```

APPENDIX E:

LOB2D SIMULATION SUMMARY

INTENTIONALLY LEFT BLANK.

APPENDIX E:

LOB2D SIMULATION SUMMARY

E-1. Introduction

This appendix provides a brief description of the "LOB2D" computer simulation model developed to provide engineering design information for Line-of-Bearing (LOB) type systems. The model was designed to provide information on both RADAR and Optical type tracking systems, however only the LOB/optical application will be discussed in this summary. The several thousand line FORTRAN source code is not included due to its length.

E-2. General

LOB2D was designed to investigate the following problem. An incoming projectile approachs a friendly target at a constant attack azimuth at a constant speed. The threat projectile can only be detected at close ranges, say 40 meters or less. It is desired that a defeat mechanism intercept and destroy the incoming threat prior to impact, at a range of 10 meters for instance. The defense tracking system's job is to track the threat from the detection point (perhaps 20 meters), continually make predictions as to WHEN and WHERE the threat will cross a 10 meter intercept line, and at some distance greater than the intercept distance (say 11 meters), make a final decision as to WHEN and WHERE intercept will occur.

Of particular interest is the accuracy required of the angular information provided by two angle-only sensors. LOB2D was designed to provide along-intercept-line location bias, along-intercept-line standard deviation information, and time-to-intercept bias and standard deviation information as a function of sensor angular variations. A wide variety of system parameters may be varied by the program inputs. The results of LOB2D calculations are intended to provide engineering guidance as to the sensor accuracy necessary to provide acceptable intercept location and time-to-intercept errors.

E-3. Simulation Model Assumptions

In the LOB2D LOB simulation model the following assumptions are made:

1. A two-dimensional model is used, i.e. threat projectile tracking is limited to the x-y plane.

2. The speed and direction-of-flight (angle-of-attack) of the threat

projectile remain constant along the flight path (due to the short distances involved).

3. Random Gaussian noise is applied to the angular information (deviations about the true sensor line-of-bearing to projectile) for each time step along the projectile flight path. The one standard deviation value for the random number generator is specified as an input and applies to both sensors (although a different random variation is applied to each).

4. The threat projectile is always detected at the input detection range and continually tracked thereafter.

5. The only noise in the simulation is applied to the sensor angular estimates as random Gaussian noise. This noise is considered to be uncorrelated.

6. The Optical sensors are simulated simply as sensors with the ability to provide angular information (with random noise) equally well within a ± 180 degree field-of-view.

7. The origin of the x-y coordinate system used in the simulation is located on the baseline defined by the two LOB sensors and is assumed to be at center front edge of the target. The sensors, although located on the baseline, need not be symmetrically located about the center of the front edge of the target.

E-4. Simulation Model Inputs

Inputs for the LOB2D simulation are normally placed in a file and redirected as standard input when the program is run. Some inputs pertain to a RADAR simulation subroutine and will be identified as such, but not discussed here. Only those inputs pertaining to the simple LOB (or optical) type sensor system will be discussed in this appendix. The threat mechanism is called either the "threat" or "projectile" and the friendly unit is called the "target". Sample inputs are listed below where the line numbers and data formats have been added for this discussion:

Line	Sample LOB2D Inputs	Input Data Format
1:	40, 10, -10	i8, i8, i8
2:	100, 0, -10	i8, i8, i8
3:	0, 0, -1	i8, i8, i8
4:	300.0, 0.0, 20.	f14.6, f14.6 f14.6
5:	10.0, 1.0, -1.0, 0.	f14.6, f14.6 f14.6, f14.6
6:	0.0005, 1	f14.8, i6
7:	1192577, 500, 0	i20, i20, i20
8:	40., 20., -20., 0.0, 1500., 1	f14.8, f14.8, f14.8, f14.8, f14.8, i6
9:	0.0	f14.6
10:	1, 1.0	i8, f14.6

Line 1 - (iaz1, iaz2, iaz3) set up a "do loop" to define the range and step size of the projectile attack azimuths to be used. An attack azimuth of 0 degrees is an attack directly from the front, +40 degrees is an attack from 40 degrees left of center, -40 degrees is an attack from 40 degrees right of center.

iaz1 - starting attack azimuth (degrees).

iaz2 - ending attack azimuth (degrees).

iaz3 - the attack azimuth step size (degrees) when moving from "iaz1" to "iaz2". A positive value for "iaz3" indicates incrementing attack azimuth, a negative value indicates decrementing the attack azimuth.

The inputs "40, 10, -10" indicate starting with an attack azimuth of 40 degrees, then decrementing in 10 degree steps to the ending attack azimuth of 10 degrees (i.e., 40, 30, 20, 10).

Line 2 - (iang1, iang2, iang3) set up a "do loop" to define the range and step size of the angular variations (one standard deviation value) to be used in the evaluation. These variations apply to the two sensors located at "d1r1" and "d2r2".

iang1 - iang1/100 is the starting angular variation (degrees).

iang2 - iang2/100 is the ending angular variation (degrees).

iang3 - iang3/100 is the angular variation step size (degrees) with steps as small as 0.01 allowed.

The inputs "100, 0, -10" indicate starting with an angular variation of 1.0 degree, then decrementing in 0.1 degree steps to the ending angular variation of 0.0 degrees.

Line 3 - (ifreq1, ifreq2, ifreq3) are "do loop" inputs for the RADAR subroutine, however, they should be set to "0, 0, -1" (which causes the loop to be ignored) when the LOB/optical simulation is run.

Line 4 - (pveloc, xxhit, ystart) define some of the projectile parameters. These parameters, along with the attack azimuth, define the projectile trajectory.

pveloc - the projectile's speed (meters/second).

xxhit - the desired projectile impact location (x-coordinate, meters) on the sensor baseline (center front edge of target is the coordinate system origin).

ystart - the y-coordinate (meters) for the start of the projectile flight. The projectile start location is determined by "ystart", "xxhit", and the current attack azimuth.

Line 5 - (py1d, d1r1, d2r2, d3t1) specify the y-locations of the intercept line and the sensors. The center front edge of the target is the coordinate system origin.

py1d - intercept line y-distance in front of target (meters).

d1r1 - x-axis distance from origin to sensor-1 (meters).

d2r2 - x-axis distance from origin to sensor-2 (meters).

d3t1 - For use by RADAR routine, ignored by LOB/optical routine.

"d1r1" is typically the right sensor with "d2r2" being the left sensor.

Line 6 - (tstep, iflag) specifies the sampling time step and sets a flag to restrict the length of flights. Flights may be stopped when the true projectile location is at either the intercept line or the impact line (sensor baseline). This flag is used for diagnostics and to guard against array overruns.

tstep - time increment for making calculations (seconds).

iflag - if 0, simulation stops at intercept line. if 1, simulation stops at impact line.

Line 7 - (ii, iter2, nwindow) random number seed, number of iterations, and averaging window size inputs.

ii - seed for random number generator (default = 0).

iter2 - the number of iterations (5000 maximum) to run a specific flight, each with a different set of random numbers drawn for the angular variations.

nwindow - RADAR parameter, ignored by LOB/optical routine.

Line 8 - (rcut, xcut1, xcut2, ycut, vmax, itest) specify the allowable limits for individual estimates of the projectiles location and speed and whether the tests should even be performed. "rcut", "xcut1", "xcut2", and "ycut" effectively define a rectangle in the x-y plane and any location estimates outside this rectangle are not acceptable and therefore discarded.

rcut - y-range (meters) beyond which y-estimate values are invalid and discarded.

xcut1 - x-distance (meters) to the right of the right sensor (d1r1) for which x-estimated impact is invalid (i.e, not a threat) and discarded.

xcut2 - x-distance (meters) to the left of the left sensor (d2r2) for which x-estimated impact is invalid (i.e., not a threat) and discarded.

ycut - minimum y-distance (meters) allowed for y-estimated projectile location (discard estimates locating the projectile behind the baseline).

vmax - maximum estimated velocity (meters per second) allowed, estimates greater than this are discarded.

itest - If greater than 0, the all the above tests are performed. If equal to 0, then no tests are performed.

Line 9 - (rngvar) is a RADAR routine input and is ignored by the LOB/optical routine.

Line 10 - (ndetect, decision) determine if a predictive run is to be made, how many samples are required, and at what y-distance prior to the intercept line the decision is to be made.

ndetect - If "ndetect" is greater than 1, then a predictive flight is performed where "ndetect" is the number of samples required before a flight is stopped because the projectile is estimated to be past the decision line. If ndetect = 0, then a flight is performed until the time step where the projectile "actually" gets to the intercept or impact line (determined by the "iflag" setting).

decision - The distance before "pyld" (intercept line) that the flight is stopped (the decision line). For instance, if "pyld" = 10.0 and "decision" = 1.5, then the sensor stops predicting when the estimated projectile location is at y = 11.5 meters. The "decision" value is only used for predictive runs, i.e. ignored if "ndetect" is less than zero.

E-5. Simulation Model Operation

The LOB2D simulation consists of a number of routines for processing flight information, checking sensor observations and estimates, performing tests on location and velocity estimates, generating random numbers, bookkeeping, etc. Of primary interest are the routines that provide the "true" information (the "precision" routine) and the estimated sensor information (the "sensor" routines). There are two "sensor" routines, one for Optical sensors and another for RADAR sensors. The desired sensor routine is selected by commenting-out the call statement to the undesired routine and compiling the program. In this report, only the Optical sensor routine was used.

In the LOB2D simulation, the projectile is first flown in a "precision" routine from initial start position to baseline impact and all the true positional, angular, range, and intercept time and location information is recorded for each time step in the flight

trajectory.

Next, a projectile flight commences (in the optical sensor routine) starting with the same initial conditions and with the "true" angular information provided for use in making the predictive calculations. For each time step in the flight, random noise is added to the true angular information and the projectile location is observed and the estimated intercept location and time-to-intercept are calculated. A recursive least squares routine (see Appendix G) is used to estimate projectile start location, current location, speed, and angle-of-attack. The projectile's observed location, estimated velocity, and estimated impact location are tested by the cutoff parameters, and if they are acceptable, the process continues. However, if any are not acceptable, then the data for the current time step are discarded and the flight continues with the next time step.

The simulation continues stepping along the flight path until the projectile reaches a point where it's estimated location is nearer than the "decision" line. This ends the flight and the final estimated intercept location and time-to-intercept are recorded.

The estimated intercept location and time-to-intercept values are next compared with the "true" intercept location and time previously calculated (in the "precision" routine) and the differences recorded.

Another flight is initiated with the same starting parameters but with different random noise applied to provide different angular variations. At the end of 500 such flights the mean and standard deviations of the intercept location differences and time-to-intercept differences were recorded and output.

Finally, the simulation increments the attack azimuth and/or angular variation parameters and initiates another set of 500 flights. This continues until all the "loop" parameters have been satisfied.

E-6. Simulation Model Sample Outputs

Figure E-1 is an example of the LOB2D outputs resulting from the inputs described in section E-4. In Figure E-4 the line numbers (on the left hand side) and the column numbers (at the bottom) have been added to aid in the discussion.

Lines 1 - 19 provide some general information and echo the input parameters:

Line 1 - documents the time and date the program was executed.

Line 2 - Blank.

Line 3 - The name of the program, i.e. LOB2D.

-->> 11:10:05, 17-Mar-92 <<--

Program "LOB2D"
iaz1= 40 iaz2= 10 iaz3= -10
iang1= 100 iang2= 0 iang3= -10
ifreq1=, 0 ifreq2= 0 ifreq3= -1
pveloc= 300.0000 xxhit= 0.0000 ystart= 20.0000
pyld= 10.00 d1r1= 1.00 d2r2= -1.00 d3t1= 0.00
tstep= 0.00050000 iflag= 1 Distance/tstep= 0.1500 meters
random number generator seed = 1192577
# interations for each set of conditions = 500
# steps in averaging window = 0 ("0" indicates all values averaged)
rcut = 40.00 xcut1 = 20.00 xcut2 = -20.00 ycut = 0.00 vmax = 1500.00 itest = 1
transmitter-to-projectile range variation (1-sigma) = 5.00% (Ignored if >= 99.0)

ndetect= 1 ("0"= non-predict, ">0"= predictive & min number samples required)
decision= 1.00 (only used if ndetect > 0)

Results for 500 iterations OPTICAL SENSOR

azi (deg)	xhit (m)	angvar (deg)	freqvar (%)	mean intcp err (m)	std dev intcp err (m)	mean time err (sec)	mean time dist (m)	std dev time err (sec)	std dev time dist (m)	number of flights (iter)
40.0	0.0000	1.0000	0.00	0.0053	0.0755	-0.0007	-0.2018	0.0037	1.0989	500
40.0	0.0000	0.9000	0.00	0.0093	0.0685	-0.0010	-0.2981	0.0030	0.8909	500
40.0	0.0000	0.8000	0.00	0.0031	0.0628	-0.0005	-0.1646	0.0037	1.0971	500
40.0	0.0000	0.7000	0.00	0.0014	0.0524	-0.0004	-0.1201	0.0032	0.9686	500
40.0	0.0000	0.6000	0.00	-0.0001	0.0464	-0.0001	-0.0416	0.0031	0.9421	500
40.0	0.0000	0.5000	0.00	0.0003	0.0391	0.0002	0.0734	0.0027	0.8215	500
40.0	0.0000	0.4000	0.00	0.0008	0.0310	0.0001	0.0238	0.0020	0.6103	500
40.0	0.0000	0.3000	0.00	-0.0002	0.0235	0.0001	0.0280	0.0016	0.4666	500
40.0	0.0000	0.2000	0.00	-0.0008	0.0157	0.0001	0.0175	0.0010	0.3004	500
40.0	0.0000	0.1000	0.00	0.0004	0.0072	0.0000	-0.0093	0.0005	0.1409	500
40.0	0.0000	0.0000	0.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	500
30.0	0.0000	1.0000	0.00	0.0001	0.0652	-0.0002	-0.0743	0.0034	1.0129	500
30.0	0.0000	0.9000	0.00	0.0047	0.0589	-0.0002	-0.0743	0.0029	0.8787	500
30.0	0.0000	0.8000	0.00	0.0002	0.0504	-0.0001	-0.0445	0.0028	0.8456	500
30.0	0.0000	0.7000	0.00	-0.0004	0.0445	0.0001	0.0302	0.0027	0.8216	500
30.0	0.0000	0.6000	0.00	0.0000	0.0404	0.0003	0.0894	0.0025	0.7517	500
30.0	0.0000	0.5000	0.00	0.0013	0.0323	0.0001	0.0155	0.0020	0.6086	500
30.0	0.0000	0.4000	0.00	-0.0019	0.0256	0.0000	0.0143	0.0015	0.4579	500
30.0	0.0000	0.3000	0.00	0.0007	0.0181	0.0000	0.0030	0.0011	0.3223	500
30.0	0.0000	0.2000	0.00	0.0004	0.0131	0.0000	-0.0147	0.0007	0.2096	500
30.0	0.0000	0.1000	0.00	-0.0002	0.0063	0.0000	-0.0001	0.0003	0.1024	500
30.0	0.0000	0.0000	0.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	500
20.0	0.0000	1.0000	0.00	0.0024	0.0583	-0.0001	-0.0441	0.0028	0.8421	500
20.0	0.0000	0.9000	0.00	0.0052	0.0494	-0.0001	-0.0279	0.0027	0.8018	500
20.0	0.0000	0.8000	0.00	0.0004	0.0442	0.0001	0.0314	0.0026	0.7844	500
20.0	0.0000	0.7000	0.00	0.0039	0.0409	0.0001	0.0313	0.0022	0.6745	500
20.0	0.0000	0.6000	0.00	0.0014	0.0321	0.0000	0.0015	0.0019	0.5751	500
20.0	0.0000	0.5000	0.00	0.0009	0.0274	0.0001	0.0402	0.0016	0.4737	500
20.0	0.0000	0.4000	0.00	-0.0002	0.0223	0.0001	0.0215	0.0012	0.3520	500
20.0	0.0000	0.3000	0.00	0.0006	0.0160	0.0000	0.0049	0.0009	0.2562	500
20.0	0.0000	0.2000	0.00	0.0002	0.0117	0.0000	0.0012	0.0006	0.1695	500
20.0	0.0000	0.1000	0.00	-0.0004	0.0051	0.0000	0.0047	0.0003	0.0828	500
20.0	0.0000	0.0000	0.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	500
10.0	0.0000	1.0000	0.00	0.0003	0.0509	0.0000	-0.0139	0.0025	0.7489	500
10.0	0.0000	0.9000	0.00	-0.0002	0.0464	0.0003	0.0879	0.0025	0.7548	500
10.0	0.0000	0.8000	0.00	-0.0001	0.0444	0.0002	0.0543	0.0022	0.6613	500
10.0	0.0000	0.7000	0.00	-0.0002	0.0362	0.0002	0.0721	0.0020	0.6085	500
10.0	0.0000	0.6000	0.00	-0.0021	0.0317	0.0002	0.0596	0.0017	0.4994	500
10.0	0.0000	0.5000	0.00	-0.0006	0.0258	0.0001	0.0297	0.0013	0.4009	500
10.0	0.0000	0.4000	0.00	-0.0003	0.0201	0.0000	0.0062	0.0010	0.3147	500
10.0	0.0000	0.3000	0.00	0.0006	0.0154	0.0000	0.0049	0.0008	0.2268	500
10.0	0.0000	0.2000	0.00	0.0000	0.0102	0.0000	-0.0058	0.0005	0.1582	500
10.0	0.0000	0.1000	0.00	0.0001	0.0050	0.0000	0.0102	0.0003	0.0761	500
10.0	0.0000	0.0000	0.00	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	500
1:	2:	3:	4:	5:	6:	7:	8:	9:	10:	11:

Figure E-1. Sample Outputs from the LOB2D Simulation

Line 4 - "iaz1", "iaz2", and "iaz3" input values.

Line 5 - "iang1", "iang2", and "iang3" input values.

Line 6 - "ifreq1", "ifreq2", and "ifreq3" input values.

Line 7 - "pveloc", "xxhit", and "ystart" input values.

Line 8 - "py1d", "d1r1", "d2r2", and "d3t1" input values. "d3t1" is only used in the RADAR simulation.

Line 9 - "tstep" and "iflag" input values and the calculated distance traveled by the projectile each time step.

Line 10 - "ii" random number generator seed input.

Line 11 - "iter" is the input number of iterations (number of flights) for each set of conditions.

Line 12 - only used by the RADAR routine.

Line 13 - "rcut", "xcut1", "xcut2", "ycut", "vmax", "itest" are the input cutoff values for estimated projectile location and velocity.

Line 14 - only used by the RADAR routine.

Line 15 - Blank.

Line 16 - "ndetect" input value.

Line 17 - "decision" input value.

Line 18 - Blank.

Line 19 - indication of the number of iterations and type of sensor (RADAR or OPTICAL) used.

Lines 21 - 25 are column labels for the output results:

Column 1 - the current attack azimuth.

Column 2 - the input projectile impact location (aim point) on the x-axis (xxhit).

Column 3 - the current one standard deviation angular variation.

Column 4 - RADAR simulation output.

Column 5 - the mean along-intercept-line difference between the true intercept location and the estimated intercept location

Column 6 - the standard deviation for along-intercept-line mean provided in column 5.

Column 7 - the mean time-to-intercept difference between the true time-to-intercept and the estimated time-to-intercept.

Column 8 - the mean time-to-intercept (column 7) multiplied by the true projectile speed to give results as range.

Column 9 - the standard deviation for the mean time-to-intercept mean provided in column 7.

Column 10 - the time-to-intercept standard deviation (column 9) multiplied by the true projectile speed to give results as range.

Column 11 - the number of iterations

Lines 26 - 69 list the output calculations. All lines following line 25 are the calculated results of the simulation runs and the number of lines depends upon the number of azimuth and/or angular variation cases desired. In Figure E-1, lines 26 through 36 give the results for a 40 degree attack azimuth with the angular variations from 1.0 to 0.0 degrees in 0.1 degree steps. Lines 37 trough 47 are the results for the 30 degree attack azimuth, lines 48 through 58 for the 20 degree attack azimuth, and lines 59 through 69 the results for the 10 degree attack azimuth.

APPENDIX F:

C AND FORTRAN RANDOM VARIABLE SUBROUTINES

INTENTIONALLY LEFT BLANK.

```
#include <math.h>
#include "ranvar.h"

void rv_gamma_drv()
/* this routine is an example of driver program that can be used to
   supply random variables for a gamma distribution
*/
{
 float x,a,b,sumx,sumx2,mu,var,dmu,diff,nx;
 int i,iseed,n;

 printf("\nEnter the number of gamma numbers desired : ");
 scanf("%f",&nx);
 printf("\nEnter the random number seed : ");
 scanf("%d",&iseed);

 n= (int) nx;
 printf("\nEnter alpha and beta : ");
 scanf("%f %f",&a,&b);

 for (i=0;i<n;i++)
 {
  x=rv_gamma(iseed,a,b);
  printf("\n%d",x);
 }
}

float rv_gamma(int iseed, float a, float b)
/* This routine generates a gamma random variable
   it is based on the acceptance rejection method
*/
{
 float g;

 if (a<1)
     g=rv_gamma_lt1(iseed, a);
 else
   if (a>1)
     g=rv_gamma_gt1(iseed,a);
   else
     g=-log(rv_ran1(iseed));
 return (g*b);
}

float rv_gamma_lt1(int iseed, float a)
/*this is used if the alpha parameter is less than one
*/
{
  float x,u1,u2,p,b,y;
  b=(a+M_E)/M_E;
  do
  {
   u1=rv_ran1(iseed);
   p=u1*b;
   if (p>1)
   {
    y=-log((b-p)/a);
    u2=rv_ran1(iseed);
    if (u2<pow(y,(a-1)))
       x=y;
      else
       x=-1;
   }

   else
```

```
    {
     y=pow(p,(1/a));
     u2=rv_ran1(iseed);
     if (u2<exp(-y))
         x=y;
       else
        x=-1;
    }

   }while (x==-1);
   return (x);
}

float rv_gamma_gt1(int iseed, float a)
/* use this section if alpha is greater than one
*/
{
 float a1,q,d,u1,u2,b,x,y,v,z,w;
 float v1,v2;

 b=a-1.386294361;      /*1.38....=ln(4) */
 a1=1/sqrt(2*a-1);
 q=a+1/a1;

 do
 {
  u1=rv_ran1(iseed);
  u2=rv_ran1(iseed);
  v1=u1/(1-u1);
  v=a1*log(v1);
  y=a*exp(v);
  z=u1*u2*u1;
  w=b+q*v-y;
  v1=w+2.504077397-4.5*z;      /*2.5040...=1+ln(4.5)   */

  if (v1>=0)
     x=y;
  else
  {
   v2=log(z);
   if (w>=v2)
       x=y;
   else
      x=-1;
  }
 }while (x==-1);
 return (x);
}
#include <math.h>
#include "ranvar.h"

float rv_cauchy(int iseed, float a, float b)
/* this returns a random realization from a cauchy distribution
*/
{
 float u,c;

 u=rv_ran1(iseed);
 c=a-b/tan(M_PI*u);
 return (c);
}

float rv_triangle(int iseed,float a,float b,float c)
/* the parameters for this triangular distribution are
   a is the lower bound
   b is the peak or mode
```

```
   c is the upper bound
*/
{
 float u,t,length,b1,b2,peak;

 length=(c-a);
 b1=b-a;
 peak=b1/length;
 u=rv_ran1(iseed);

 if (u>peak)
 {
  b2=(c-b);
  t=c-sqrt((1-u)*b2*length);
 }
 else
 {
  t=sqrt(u*b1*length)+a;
 }
 return (t);
}

float rv_a1(int iseed, int stream, float decay)

/* this routine returns a number from an exponentially auto correlated
    noise process.
   There are ten different streams of these autocorrelated gaussian variables.
   This can be increased by raising the dimension of the static variables

*/
{
 static float lastx[10],bt[10],error_mag[10];
 static int first=0, flag[10];
 float v1, dx;
 int i;

 if (first==0)
  {
    for (i=0;i<10;i++)
     flag[stream]=0;
    first=1;
   }

 if (flag[stream]==0)
 {
  bt[stream]=exp(-decay);
  error_mag[stream]=sqrt(2*bt[stream])*(1-bt[stream]);
  flag[stream]=1;
  lastx[stream]=rv_gauss(iseed);
  for (i=1;i<100;i++)
    lastx[stream]=bt[stream]*lastx[stream]+rv_gauss(iseed)*error_mag[stream];
 }

 v1=rv_gauss(iseed);
 lastx[stream]=bt[stream]*lastx[stream]+v1*error_mag[stream];
 return(lastx[stream]);
}

float rv_a2(int iseed, int stream, float decay)
/* this routine simulates an second order autoregressive process*/
{
 static float xt_1[10],xt_2[10],b1[10],b2[10];
 static int flag[10],first=0;
 float v1,xt;
 int i;
```

```
 if (first==0)
 {
  for (i=0;i<10;i++)
    flag[i]=0;
  first=1;
 }
 if (flag[stream]==0)
  {
   b1[stream]=decay;
   b2[stream]=decay*decay;
   flag[stream]=1;
   xt_1[stream]=rv_gauss(iseed);
   xt_2[stream]=xt_1[stream]*decay+rv_gauss(iseed);
  }
  xt=xt_1[stream]*b1[stream]+xt_2[stream]*b2[stream]+rv_gauss(iseed);
  xt_2[stream]=xt_1[stream];
  xt_1[stream]=xt;
  return(xt);
}

float rv_ar2_2(int iseed,int stream,float time,float b1,float b2)

{
 static int flag[10], first=0;
 static float x1[10], x2[10],m11[10],m12[10],m22[10],g[10];
 int i;

 if (first==0)
 {
  for (i=0;i<10; i++)
   flag[i]=0;
  first=1;
 }

 if (flag[stream]==0)  /*If this stream of numbers has not been initialized*/
 {
  flag[stream]=1;     /* set the initialization indicator*/

  if (b1==b2)         /* if both decays are equal do the following */
  {
    m11[stream]=exp(-b1*time);   /* the m variables hold the transition*/
    m22[stream]=m11[stream];     /* matrix */
    m12[stream]=time*m11[stream];
  }
   else      /* the decay rates are not equal set the transition matrix */
  {
   m11[stream]=exp(-b1*time);
   m22[stream]=exp(-b2*time);
   m12[stream]=(m11[stream]-m22[stream])/(b1-b2)*sqrt(2*b1);
  }
/* intialize the state variables */
  x1[stream]=rv_gauss(iseed);
  x2[stream]=rv_gauss(iseed);
  g[stream]=sqrt(2*b2)*(1-m22[stream]);  /* the g variable sets the scale of th
                          /*for the amount of time that passes*/

  for (i=1;i<100;i++)   /* let the state propagate for a while */
  {
   x1[stream]=m11[stream]*x1[stream]+m12[stream]*x2[stream];
   x2[stream]=m22[stream]*x2[stream]+g[stream]*rv_gauss(iseed);
  }
 }

/* this is only part executed after the initialization and is the
   error propagation*/
 x1[stream]=m11[stream]*x1[stream]+m12[stream]*x2[stream];
```

```
 x2[stream]=m22[stream]*x2[stream]+g[stream]*rv_gauss(iseed);
 return(x1[stream]);
}

 float rv_ran1(int seed)
 /* this random number generator is based on the suggestions for improvement
 from numerical recipies */

{
 static float y,maxran,v[98];
 float dum;
 static int iff=0;
 int j;
 unsigned i,k;

 if (seed<0|| iff==0)
 {
  iff=1;
  i=2;
  do (k=i;i<<=1;) while (i);
  maxran=k;
  srand(seed);
  seed=1;
  for (j=1;j<197;j++) dum=rand();
  for (j=1;j<98;j++) v[j]=rand();
  y=rand();
 }
j=1+97.0*y/maxran;
y=v[j];
v[j]=rand();
return (y/maxran);
}

float rv_gauss (int seed)
/* this is a normal random number generator */
{
 static int iset=0;
 static float gset;
 float fac,r,v1,v2;

 if (iset==0)
 {
  do
   {
    v1=2.0*rv_ran1(seed)-1.0;
    v2=2.0*rv_ran1(seed)-1.0;
    r=v1*v1+v2*v2;
   } while (r>=1.0);
  fac=sqrt(-2.0*log(r)/r);
  gset=v1*fac;
  iset=1;
  return (v2*fac);
 }
 else
 {
  iset=0;
  return gset;
 }
}

float rv_ermav5x(int seed)
/* this routine generates a moving average process of length five
   the most recent values have the greatest weight
   it should be easy to alter this for a different weighted average
   noise process
*/
```

```
{
 static float w[5],g[5];
 static int flag=0;
 int i;
 float res=0;

 if (flag == 0)
 {
  w[0]=.333;                              /*these are the weights*/
  w[1]=.25;
  w[2]=.167;
  w[3]=.167;
  w[4]=.083;
  for(i=0;i<5;i++)
   {
    g[i]=rv_gauss(seed);
    res=w[i]*g[i];
   }
  return res;
 }
 else
 for (i=4;i>0;i--)
  g[i]=g[i-1];
 g[0]=rv_gauss(seed);
 res=0;
 for (i=1;i<5;i++)
  res+=g[i]*w[i];
 return res;
}

float rv_weibel(int iseed, float a, float b)
/* call this to get an rv form the weibel distribution
*/
{
 float x,v1,v2,u1;

 u1=rv_ran1(iseed);
 v1=-log(u1);
 v2=pow(v1,(1/a));
 x=b*v2;
 return (x);
}

float rv_chi_sq(int iseed, float df)
/*this generates a chi square random variable
*/
{
 float xsq,a,b;

 b=2;
 a=df/2;
 xsq=rv_gamma(iseed, a, b);
 return (xsq);
}

float rv_t(int iseed, float df)
/*this generates a random variable from the t distribution
*/
{
 float t,n,c;
 n=rv_gauss(iseed);
 c=rv_chi_sq(iseed,df);
 t=n/sqrt(c/df);
 return (t);
}
```

```
float rv_f(int iseed, float df1, float df2)
/*this generates a random variable from an f distribution
*/
{
 float c1,c2,f;
 c1=rv_chi_sq(iseed,df1);
 c2=rv_chi_sq(iseed,df2);
 f=(c1*df2)/(c2*df1);
 return (f);
}

float rv_normal(int iseed, float mu, float var)
/*included for generating a normal random variable with a non zero mean
  mu and a variance of var
*/
{
 float sd,n;
 sd=sqrt(var);
 n=mu+sd*rv_gauss(iseed);
 return (n);
}

float rv_log_normal(int iseed, float mu, float var)
/* call this to get a lognormal random diviate
*/
{
 float y;
 y=exp(rv_normal(iseed,mu,var));
 return (y);
}

float rv_beta(int iseed, float a1,float a2)
/*a random varible form a beta distribution is simulated by this routine
*/
{
 float x,y1,y2;

 y1=rv_gamma(iseed,a1,1);
 y2=rv_gamma(iseed,a2,1);
 x=y1/(y1+y2);
 return (x);
}
c
c    >>>>>    Some Fortran Random Variable Subroutines   <<<<<


c
      subroutine beta(a1,a2,iseed,x)
         b=1
         call gamma(a1,b,iseed,x1)
         call gamma(a2,b,iseed,x2)
       x=x1/(x1+x2)
       end

c************************************
c EXPONENTIAL RV Subroutine
c************************************
c
       subroutine expo(iseed,mu,ex)
       ex=-mu*log(ran(iseed))
       return
       end
c************************************
c GAMMA RV subroutine
c************************************
c
```

```
      subroutine gamma(a,b,iseed,x)
c choose corect section
      if (a.lt.1) goto 10
      q=(a+1)/a
      b1=a-1.3862944
      a=(2*a-1)**.5
  1   r1=ran(iseed)
      r2=ran(iseed)
      v=a*log(r1/(1-r1))
      y=a*exp(v)
      z=r1**2*r2
      w=b1+q*v-y
      if ((w+1.5040774-4.5*z).ge.0) goto 100
      if (w.ge.log(z)) goto 100
      goto 1
 10   b1=(2.7182818+a)/2.7182818
 11   r1=ran(iseed)
      p=b1*r1
      if (p.gt.1) goto 13
      y=p**(1/a)
      r2=ran(iseed)
      if (r2.gt.(exp(-y))) goto 11
      goto 100
 13   y=-log((b1-p)/a)
      r2=ran(iseed)
      if (r2.gt.(y**(a-1))) goto 11
100   x=y*b
      end

c ********************************************************
c function to produce two gaussian variables g1,g2
c ********************************************************
c
      subroutine gauss(iseed,g1,g2)
    5 u=2*ran(iseed)-1
      v=2*ran(iseed)-1
      r=u*u+v*v
      if (r.ge.1) goto 5
      s=sqrt(-2*alog(r)/r)
      g1=u*s
      g2=v*s
      end

c ***************************************************************
c subroutine for a triangular distribution
c a is lowest value b is the mode or peak c is the greatest value
c***************************************************************
      subroutine triangular(a,b,c,val)
      common /seed/ iseed
      real num1
      r=ran(iseed)
      d=2/(c-a)
      tst=.5*d*(b-a)
      if (r.lt.tst) then
         num1=(8*(b-a)*r/d)**.5
         val=(2*a+num1)/2
         goto 10
       else
         num1=(8*(1-r)*(c-b)/d)**.5
         val=(2*c-num1)/2
      endif
10    return
      end

c ***************************************************************************
c  this subroutine provides the same (0,1) random number stream on
```

```
c  all 32 bit machines
c ***********************************************************************
      subroutine gauss(iseed,v1,v2)
      implicit real (a-z)
5     u=2*uran31(iseed)-1
      v=2*uran31(iseed)-1
      r=u*u+v*v
      if (r.ge.1) goto 5
      s=sqrt(-2*alog(r)/r)
      v1=u*s
      v2=v*s
      return
      end
      function uran31(irand)

      modulus=67108864

      if (irand.eq.0) irand=11111111
      j=irand
      j=irand*25
      j=j-int(j/modulus)*modulus
      j=j*25
      j=j-int(j/modulus)*modulus
      j=5*j
      j=j-int(j/modulus)*modulus
      ai=float(j)
      irand=j
      uran31=ai/67108864
      return
      end
```

INTENTIONALLY LEFT BLANK.

APPENDIX G:

LEAST SQUARES ESTIMATION TECHNIQUES

Appendix G

This appendix describes some estimation techniques for two situations. In the first problem the location of a stationary target is estimated from more than one fix. The second problem is to estimate the trajectory of a projectile from a series of cuts. Estimators can be classified by a criteria or cost metric they use, and by a method of data processing. Typical criteria include minimizing the sum of squares, minimizing the predictive error, and maximizing the likelihood of a conditional joint density function; least squares estimates LSE, predictive error estimates PEE, and maximum likelihood estimates MLE result from each criteria. If the estimator processes all the data in one pass and then gives the estimate this is batch processing. Recursive estimators process the data one point at a time and update the estimate and the covariance of the estimate on each cycle. Iterative processing involves making more than one pass through the data stopping when the change in the estimate from one pass to the next falls beneath a predefined threshold. Typically, LOB information is arriving sequentially and for that reason recursive methods are preferred. For more detailed accounts see Kumar and Varaiya, Spriet and Vansteenkiste, Maybeck, Gelb, or Widrow and Sterns.

Recursive processing updates the estimate based on a pairwise combination of the current value and the new observation. The variance of the estimate is also updated during each estimate upgrade. Typically the first observation is used to start the recursive process. In other cases the process can be started by either a combination of the first several observations or by guessing the first value and associating low confidence with the guess. As each new observation becomes available for processing a gradient is added to the current estimate, this result becomes the current estimate. The gradient consists of two components; first the difference between the observation and the current estimate, and second the relative value of the observation. The value of the observation is based on the covariance of the current estimate and the covariance of the observation.

The notation for recursive estimation comes from control theory, not from statistics. To update the estimate four pieces of information are needed:

1. The current estimate X^-
2. The covariance of the current estimate P^-
3. The current observation Z
4. The covariance of the current observation R

In formal discussions recursive estimators are frequently discussed in terms of the updated estimate X^+ and its covariance P^+. They are calculated as follows.

$$X^+ = X^- + P^-(P^- + R)^{-1}(Z - X^-) \tag{1}$$

$$P^+ = P^- - P^-(P^- + R)^{-1}P^- \tag{2}$$

It can also be shown that in the recursive least squares case formula (2) is the same as

$$P^+ = ((P^-)^{-1} + R^{-1})^{-1} \tag{3}$$

If the errors are independently, identically distributed the following formulation of recursive estimation for LSE is preferred.

$$A^+ = A^- + H_i X_i \tag{4}$$

$$P^+ = ((P^-)^{-1} + H_i H'_i)^{-1} \tag{5}$$

$$X^+ = P^+ A^+ \tag{6}$$

In the above equations H_i is a vector of the values of the independent variables. A^+ and A^- are intermediate results that measure the projection of the dependent variable onto the independent variables.

An example will be worked out to show how to use the above equations. This example demonstrates the method for recursive estimation of a location in the x-y plane.

$$\text{Let } X^- = \begin{pmatrix} 100 \\ 100 \end{pmatrix}, \text{ and } P^- = \text{Cov}(X^-) = \begin{bmatrix} 10 & 5 \\ 5 & 30 \end{bmatrix}$$

The correlation between successive estimates is assumed to be negligible.

$$\text{Let the new observation be } Z = \begin{pmatrix} 95 \\ 110 \end{pmatrix}, \text{ with covariance matrix } R = \text{Cov}(Z) = \begin{bmatrix} 15 & 10 \\ 10 & 15 \end{bmatrix}.$$

Using equation 1 the new estimate X^+ is:

$$\begin{aligned} X^+ &= \begin{pmatrix} 100 \\ 100 \end{pmatrix} + \begin{bmatrix} 10 & 5 \\ 5 & 30 \end{bmatrix} \left[\left[\begin{bmatrix} 10 & 5 \\ 5 & 30 \end{bmatrix} + \begin{bmatrix} 15 & 10 \\ 10 & 15 \end{bmatrix} \right]^{-1} \left[\begin{pmatrix} 95 \\ 110 \end{pmatrix} - \begin{pmatrix} 100 \\ 100 \end{pmatrix} \right] \right] \\ &= \begin{pmatrix} 100 \\ 100 \end{pmatrix} + \begin{pmatrix} 10 & 5 \\ 5 & 30 \end{pmatrix} \begin{pmatrix} 45 & -15 \\ -15 & 25 \end{pmatrix} \frac{1}{900} \begin{pmatrix} -5 \\ 10 \end{pmatrix} \\ &= \begin{pmatrix} 97.6389 \\ 108.75 \end{pmatrix} \end{aligned}$$

The update of the covariance is found through equation 2 or equation 3. The resulting variance of the estimate using equation 2 is

$$\begin{aligned} P^+ &= \begin{bmatrix} 10 & 5 \\ 5 & 30 \end{bmatrix} - \begin{bmatrix} 10 & 5 \\ 5 & 30 \end{bmatrix} \left[\left[\begin{bmatrix} 10 & 5 \\ 5 & 30 \end{bmatrix} + \begin{bmatrix} 15 & 10 \\ 10 & 15 \end{bmatrix} \right]^{-1} \begin{bmatrix} 10 & 5 \\ 5 & 30 \end{bmatrix} \right] \\ &= \begin{bmatrix} 10 & 5 \\ 5 & 30 \end{bmatrix} - \begin{bmatrix} 10 & 5 \\ 5 & 30 \end{bmatrix} \frac{1}{900} \left[\begin{bmatrix} 45 & -15 \\ -15 & 25 \end{bmatrix} \begin{bmatrix} 10 & 5 \\ 5 & 30 \end{bmatrix} \right] \\ &= \frac{1}{36} \begin{bmatrix} 215 & 135 \\ 135 & 315 \end{bmatrix} \end{aligned}$$

Using equation 3 gives the following expression for finding the covariance

$$P^+ = \left[\left[\begin{bmatrix} 10 & 5 \\ 5 & 30 \end{bmatrix}^{-1} + \begin{bmatrix} 15 & 10 \\ 10 & 15 \end{bmatrix}^{-1} \right]^{-1} \right]$$

$$= \left[\frac{1}{275} \begin{bmatrix} 30 & -5 \\ -5 & 10 \end{bmatrix} + \frac{1}{125} \begin{bmatrix} 15 & -10 \\ -10 & 15 \end{bmatrix} \right]^{-1}$$

$$= \frac{1}{36} \begin{bmatrix} 215 & 135 \\ 135 & 315 \end{bmatrix}$$

When the next observation, Z, arrives the current X^+ and P^+ become X^- and P^- and the process continues until a decision is made.

In the preceding example the covariance of each estimate was known. This is not usually the case. Typically the covariance must be estimated. A quick method is to derive the covariance as if the estimate of the target location is exact. This gives adequate results in most cases. Usually simulations can be used to test if this approach is appropriate for a specific situation. Another method is to use an iterative approach. On the first pass average the values for the location. On the second pass, process all the data using the location derived in the first pass to calculate the covariances of each location estimate. On the third pass, use the location that resulted from the second pass to derive the covariance of the estimates. This process would be repeated until the change in the estimated location falls below some threshold. Unfortunately it is not always possible to use iterative techniques while processing data. Many times methods will be developed that try to approach iterative techniques while recursively processing the data in one pass. The performance of a specific system can be verified through a simulation.

For the remainder of this appendix, the problem of estimating the trajectory of a projectile will be considered. It will be assumed that the time a fix is acquired is known without error; time is considered an independent variable. A quadratic trajectory could be modeled by including t_i^2 as an independent variable, but this is not pursued in the sequel. The assumptions made are as follows.

1. The trajectory is best described by a straight line.
2. The problem can be broken into two independent problems; the estimation of the trajectory along the X axis, and the estimate of the trajectory along the Y axis.

The following two equations are the models that will be used for the X and Y data.

$$X_i = \alpha_0 + \alpha_1 t_i \tag{7}$$

$$Y_i = \beta_0 + \beta_1 t_i \tag{8}$$

The time of the first observation, t_1, is taken to be 0; and each of the subsequent t_i's is the time duration from the initial observation. Each of the independent variables H_i will be $(1, t_i)'$. The parameters sets, $\{\alpha_0, \alpha_1\}$ and $\{\beta_0, \beta_1\}$ will each have a covariance matrix of P which can be found through the following relationships.

$$P^{-1}=\sum_{i=1}^{N} H_i H'_i = \begin{bmatrix} N & \sum_{i=1}^{N} t_i \\ \sum_{i=1}^{N} t_i & \sum_{i=1}^{N} t_i^2 \end{bmatrix} \tag{9}$$

In this two dimensional problem finding the inverse is not a problem; however for higher order equations the matrix inversion lemma can be used to reduce the computational load (see one of the references for more details). Equations 4-6 give the most straightforward estimation technique for this situation. The assumptions are the observations are independently identically distributed. If the variance of the measurement noise varies from observation to observation these equations can be generalized to the following:

$$A^+ = A^- + H_i R^{-1}{}_i X_i \tag{10}$$

$$P^+ = ((P^-)^{-1} + H_i R^{-1}{}_i H'_i)^{-1} \tag{11}$$

$$X^+ = P^+ A^+ \tag{12}$$

Assuming the observations are uncorrelated and that $\sigma_{y_i}{}^2$ is the variance associated with the ith observation of the y location then the variance of the parameters, (β_0, β_1), after n observations will be

$$P^{-1}=\sum_{i=1}^{N} H_i H'_i = \begin{bmatrix} \sum_{i=1}^{N} \sigma_{y_i}{}^{-2} & \sum_{i=1}^{N} t_i \sigma_{y_i}{}^{-2} \\ \sum_{i=1}^{N} t_i \sigma_{y_i}{}^{-2} & \sum_{i=1}^{N} t_i{}^2 \sigma_{y_i}{}^{-2} \end{bmatrix} \tag{13}$$

The variance of the parameters associated with the motion along the X axis can be found by using the X measurement variance in place of those for Y in equation 13. Equations 9 or 13 can be used to define the optimal lower bound on system performance. Various techniques can be compared to these bounds to indicate how much more improvement is possible.

When the observations are correlated more complex methods must be used if it is necessary for the estimator to consider the effects of the correlation. In this situation the least squares criteria is not adequate. Methods include instrumental variables and approximate maximum likelihood. Adaptive filters are used when the parameters change over time; sometimes it is possible to include specific knowledge about a systems dynamics within the filter. The reader is referred to the references for additional information.

INTENTIONALLY LEFT BLANK.

No. of Copies	Organization
2	Administrator Defense Technical Info Center ATTN: DTIC-DDA Cameron Station Alexandria, VA 22304-6145
1	Commander U.S. Army Materiel Command ATTN: AMCAM 5001 Eisenhower Ave. Alexandria, VA 22333-0001
1	Commander U.S. Army Laboratory Command ATTN: AMSLC-DL 2800 Powder Mill Rd. Adelphi, MD 20783-1145
2	Commander U.S. Army Armament Research, Development, and Engineering Center ATTN: SMCAR-IMI-I Picatinny Arsenal, NJ 07806-5000
2	Commander U.S. Army Armament Research, Development, and Engineering Center ATTN: SMCAR-TDC Picatinny Arsenal, NJ 07806-5000
1	Director Benet Weapons Laboratory U.S. Army Armament Research, Development, and Engineering Center ATTN: SMCAR-CCB-TL Watervliet, NY 12189-4050
(Unclass. only)1	Commander U.S. Army Rock Island Arsenal ATTN: SMCRI-TL/Technical Library Rock Island, IL 61299-5000
1	Director U.S. Army Aviation Research and Technology Activity ATTN: SAVRT-R (Library) M/S 219-3 Ames Research Center Moffett Field, CA 94035-1000
1	Commander U.S. Army Missile Command ATTN: AMSMI-RD-CS-R (DOC) Redstone Arsenal, AL 35898-5010

No. of Copies	Organization
1	Commander U.S. Army Tank-Automotive Command ATTN: ASQNC-TAC-DIT (Technical Information Center) Warren, MI 48397-5000
1	Director U.S. Army TRADOC Analysis Command ATTN: ATRC-WSR White Sands Missile Range, NM 88002-5502
1	Commandant U.S. Army Field Artillery School ATTN: ATSF-CSI Ft. Sill, OK 73503-5000
(Class. only)1	Commandant U.S. Army Infantry School ATTN: ATSH-CD (Security Mgr.) Fort Benning, GA 31905-5660
(Unclass. only)1	Commandant U.S. Army Infantry School ATTN: ATSH-CD-CSO-OR Fort Benning, GA 31905-5660
1	WL/MNOI Eglin AFB, FL 32542-5000
	Aberdeen Proving Ground
2	Dir, USAMSAA ATTN: AMXSY-D AMXSY-MP, H. Cohen
1	Cdr, USATECOM ATTN: AMSTE-TC
3	Cdr, CRDEC, AMCCOM ATTN: SMCCR-RSP-A SMCCR-MU SMCCR-MSI
1	Dir, VLAMO ATTN: AMSLC-VL-D
10	Dir, USABRL ATTN: SLCBR-DD-T

No. of
Copies Organization

1 UMBC
Dept. of Mathematics and Statistics
ATTN: Bimal K. Sinha
5401 Wilkens Ave.
Catonsville, MD 21228

2 Director
NSA
E51 ATTN: Charles Alexander
9800 Savage Road
Fort Meade, MD 20755

Aberdeen Proving Ground

7 Dir, USAMSAA
ATTN: AMXSY-GA, Bill Yeakel
AMXSY-RM, Woodworth
AMXSY-CA, O'Neil
AMXSY-CA, Clay
AMXSY-A, Hennessy
AMXSY-GA
AMXSY-GS
AMXSY-CC, Sandmeyer

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. BRL Report Number BRL-TR-3390 Date of Report September 1992

2. Date Report Received _____

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. _____

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) _____

CURRENT ADDRESS

Name

Organization

Address

City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the New or Correct Address in Block 6 above and the Old or Incorrect address below.

OLD ADDRESS

Name

Organization

Address

City, State, Zip Code

(Remove this sheet, fold as indicated, staple or tape closed, and mail.)

DEPARTMENT OF THE ARMY
Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

OFFICIAL BUSINESS

BUSINESS REPLY MAIL
FIRST CLASS PERMIT No 0001, APG, MD

Postage will be paid by addressee.

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES